ASD-TDR-62-265
PART II

AN ADAPTIVE INFORMATION TRANSMISSION SYSTEM EMPLOYING MINIMUM REDUNDANCY WORD CODES

TECHNICAL DOCUMENTARY REPORT ASD-TDR-62-265, PART II

Eugene S. Schwartz

1 JUNE 1963

Electromagnetic Warfare and Communications Laboratory
Aeronautical Systems Division
Air Force Systems Command
Wright-Patterson Air Force Base, Ohio

Project No. 4335

Task No. 433519

Prepared under Contract No. AF 33(616)-7882

Armour Research Foundation
of
Illinois Institute of Technology
Technology Center
Chicago 16, Illinois

NOTICES

FOREWORD

This report was prepared by Armour Research Foundation of Illinois Institute of Technology, Chicago, Illinois on Air Force contract AF 33(616)-7882, under Task No. 433510 of Project No. 4335, "Study of Basic English System". The work was administered under the direction of the Electromagnetic Warfare and Communications Laboratory, Aeronautical Systems Division.

The report is the final report for Phases 1 and 2 of research which was performed in the period 1 February 1961 to 1 April 1963. The main body of the report is concerned with research on Phase 2 conducted in the second year of the program. The report on Phase 1 which contains the theoretical background of the adaptive information transmission system was published as Technical Documentary Report ASD-TDR-62-265, 1 April 1962. This was not designated Part I.

The following members of the research staff have made significant contributions to the development of the system described in this report:

Bruce Kallick prepared the basic flow charts of the encoding routine.

Antanas V. Dundsila programmed the encoding, code assignment, and code breakpoint table routines, and prepared the transmitter and receiver dictionary tapes.

Adrienne J. Lipson programmed the word analysis routines.

Arthur Wachowski programmed the transmitter enciphering routines and prepared the section on System Organization.

Janis J. Pettyjohn programmed the receiver routines and the error detection system for both the transmitter and receiver.

Leon Wester contributed to the analysis of bottom merging and dictionary adaptation.

ABSTRACT

An experimental model of an adaptive information transmission system employing minimum redundancy word codes is described. Flow charts of computer programs for generating a canonical encoding, preparing transmitter and receiver dictionary tapes, adapting the dictionary to usage, and for enciphering, deciphering, and decoding messages are presented. Compression ratios for processed messages were determined and [illegible]iveness of a modified split dictionary format providing for word decomposition and synthesis was studied. It is shown how knowledge concerning the statistics of word and letter frequencies can be effectively utilized in the design of an efficient system. A procedure for determining when dictionary changes are required is outlined.

Characteristics of exhaustive prefix encodings are investigated and a proof is given that bottom merging in the Huffman encoding minimizes $\Sigma\, p_i L_i$, $\Sigma\, L_i$, and L_{max}. The number of exhaustive prefix encodings for a given maximum code length, the distribution of the number of codes in these encodings, and the non-uniqueness of optimal encodings are discussed.

An error-detection system based upon the property of self-synchronization of variable length non-spaced prefix encodings is presented and experimental results are given. Procedures for determining a synchronizing sequence are developed and a system for detecting errors involving any number of digits in any location is described. The error detecting capabilities of the system are reviewed.

TABLE OF CONTENTS

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF FIGURES

LIST OF TABLES

LIST OF TABLES

LIST OF FLOW CHARTS

LIST OF FLOW CHARTS

I. INTRODUCTION

Significant advances in information transmission systems have resulted from analysis of the properties of communication channels and transmitted signals. Statistical properties of the information source, however, have usually been ignored, especially in transmission of natural languages. The integration of computers as active elements in communications systems affords opportunities for fuller utilization of the insights of information theory and symbol manipulation techniques. Removal of the redundancy inherent in natural languages through compression by encoding and reintroduction of a measured quantity of programmed redundancy for error detection and correction is made possible. In addition, computers can serve as adaptive elements in a dynamic system and as tools for analysis of the communications process.

An adaptive information transmission system based upon the above concepts was developed under Phase I of this research program and has been described in Technical Documentary Report ASD-TDR-62-265, 1 April 1962. The present report describes the work performed during Phase 2 and includes detailed results of tests performed with the experimental system, linguistic analyses, and further studies on the characteristics of exhaustive prefix encodings. In addition, an error detection system based upon the property of self-synchronization of non-spaced variable length prefix encodings is described using both theoretical and experimental analysis.

The major objective of the research program is to increase the efficiency of information transmission systems by reducing the number of

Manuscript released by the author 1 April 1963 for publication as an ASD Technical Documentary Report.

binary digits required in transmitting messages. After studying abbreviation systems, synthetic languages, condensed vocabularies, and format systems, it was determined that none of these language compression techniques could satisfy the criteria for an efficient system with minimum cost and possessing the flexibility and adaptibility required.

Language compression in an information-theoretic sense was found to afford lowest cost while permitting use of a natural language without artificial constraints. Accordingly, an information transmission system employing minimum redundancy word codes was selected as best meeting the requirement to reduce redundancy. The system has the advantage that natural English is the input and output language. No restrictions are therefore placed on vocabulary and standard rules of grammar and syntax are employed. Translation is a straightforward enciphering-deciphering process with computers performing all the clerical, search, and translation operations. The system is, in the main, independent of any specific corpus of words or their frequency count and has been designed to operate using either the a priori or the a posteriori statistics of any information source.

This report has incorporated portions of the report on Phase 1 in those sections where subsequent studies have extended the findings of the earlier report and it was found desirable to present an integrated discussion. In the main, the first report should be considered as a companion report and the detailed theoretical and analytical results it contains have not been repeated in the present report. Where required, earlier findings have been summarized and appropriate references are indicated. The first report on Phase 1 is reference [1] and pages are indicated following the colon, as in [1:17] .

This report consists of 11 major sections. Section II is a general description of the adaptive information transmission system and includes simplified system flow charts, an analysis of the modified split dictionary format that is employed, and illustrates a simulated transmission of a message through the transmitter computer and the receiver computer.

The experimental dictionary is discussed in Section III and is, in general, a summary of the studies of Phase 1. Theoretical analysis of the word decomposition-synthesis programs is presented together with dictionary statistics.

In Section IV, the characteristics of exhaustive prefix encodings are examined and the generation of an optimum encoding is described. A proof that the bottom merging procedure produces an encoding that minimizes $\sum p_i L_i$, $\sum L_i$, and L_{max} is presented. The canonical encoding generated for the 5,208 entries of the experimental dictionary is described. In addition, the number of exhaustive prefix encodings for a given maximum code length, the distribution of the number of codes in these encodings, and the non-uniqueness of optimal encodings are investigated.

The compression ratio, effectiveness of the word synthesis program, and translation accuracy obtained from processing various texts are discussed in Section V.

Problems associated with the adaptive features of the system are considered in Section VI. The accretion rate of a growing dictionary, linguistic analysis, and convergence of a growing dictionary to an optimum encoding are presented.

System organization, processing rates, and economic considerations are surveyed in Section VII.

A detailed description of an error detection system employing the property of self-synchronization of variable length non-spaced prefix encodings is contained in Section VIII.

Detailed, machine-independent flow charts of the computer programs make up Section IX. A description accompanies each flow chart.

Conclusions are contained in Section X and references are listed in the bibliography of Section XI.

II. THE ADAPTIVE INFORMATION TRANSMISSION SYSTEM

The adaptive information transmission system utilizes a computer at the information source to compute statistics of the source, assign codes, encipher messages, introduce programmed redundancy for error checking, and adapt dictionaries and their encoding in accordance with usage. A computer at the receiver deciphers a string of binary digits, performs error checks, and decodes to recover the original message. The system is flexible for use with restricted or unrestricted storage capabilities. A static or growing dictionary can be used in conjunction with a general glossary, a microglossary, or several microglossaries, each stored on a separate tape with its distinct encoding. The system can utilize an integrated dictionary in which each word type is separately listed, or a split dictionary from which paradigmatic forms can be synthesized from a list of stems and affixes.

Data for statistical analysis are obtained from: cumulative counters for words, letters, numerals, and other symbols processed in input text; counters for word, numeral, and symbol frequencies; and counters for frequencies of unlisted words, number of letters in these words, number of complex words synthesized under the split dictionary option, and total number of binary digits transmitted.

The system operates in two modes: letter and word. In letter mode, words not appearing in the dictionary are spelled out using minimum redundancy letter codes and the words are stored on a "new word" tape. In initial operation without a prepared dictionary, all transmission will be in letter mode. After transmitting a given number of words, the "new-word" tape is inventoried and codes can be assigned to the frequency-sorted word list.

In subsequent periods, operation will be bimodal. Codes of input words listed in the dictionary are transmitted and counters are incremented; unlisted words are transmitted in letter mode. At the end of processing a specified number of words, new words are merged in a revised list according to adjusted frequencies and codes are reassigned. Additional new words can be accreted or a designated number of most frequent words can be selected, depending upon storage capabilities.

Given either the a priori or the a posteriori statistics of the information source, a dictionary is encoded by the method described in Section IV. B. Enciphering a message produces a string of binary digits consisting of the code for the first word, followed by the code for the second word, etc. If the encoding has the prefix property such that no word is encoded in a way that its sequence of digits appears as the initial digits of any code of greater length, an enciphered message will be uniquely decipherable.

Deciphering a message entails decomposition of a string of binary digits into its constituent codes and the association of each code with the word it represents. The digit string is decomposed by examining the incoming digits, one by one, and determining from a breakpoint table, the number of digits required for a code with a given binary value. If the entire dictionary can be stored in the computer, it is necessary to store only the words represented by the codes and decoding takes place by determining the storage location of the word.

The problems of dictionary size and format were extensively reviewed in [1:45-83] . Since it is impossible to include all words in the English language in the system, it is necessary to select a restricted corpus of words which will not overtax the storage capacity of a computer. It is required at the same time, however, that the words selected will permit efficient transmission considering that non-dictionary words must also be transmitted. Transmission of non-dictionary words will entail extra cost as they will ordinarily have to be spelled out letter by letter.

Resolution of the problem of restricted dictionary size while maintaining efficient transmission by minimizing the number of binary digits in a message at minimum processing cost is met by judicious choice of a dictionary format. The format poses a number of complex problems because of the presence of inflectional, derivational, and composition word forms.

A dictionary can be arranged as one integrated word list in which unique forms of a word (types) appear as separate words. Thus, in an integrated dictionary, the following words might appear:

work	worker	reworked
works	workers	reworking
worked	rework	workable
working	reworks	unworkable

An alternative arrangement is to employ a split dictionary in which word stems appear together with a list of affixes. The stems may be either free forms, that is, symbol strings which can be used alone with meaning as the word <u>work</u>, or bound forms such as <u>cept</u> in the word <u>concept</u> or <u>abund</u> in the word <u>abundance</u>. Just as the paradigmatic set of words listed above can be formed with the free form <u>work</u> so sets of words can be formed with the bound form: <u>concept</u>, <u>incept</u>, <u>recept</u>, and <u>abundant</u>, <u>abundance</u>.

The affixes may be either prefixes as in <u>re-work</u> and <u>con-cept</u>, or suffixes as in <u>work-ed</u> and <u>abund-ance</u>. The affixes may be single-layered as in the examples of the preceding sentence or multi-layered as in <u>work-er-s</u>, <u>un-work-able</u>, <u>ex-cept-ion-al-ly</u>.

Even though it is theoretically possible to decompose a word into a number of constituent elements in accordance with given decision rules based upon etymological, syntactical, and grammatical analysis, the cost of complex word decomposition and word synthesis routines can become prohibitive. Accordingly, in [1] the problem of organizing a "best" type of a split dictionary was studied.

Considering only free forms, studies have shown that between 40 and 50 percent of words in text appear in other than canonical form, i. e., the infinitive form of a verb without "to", the singular form of nouns, and the positive degree of adjectives and adverbs [1:58-60] .

A split dictionary permits use of a language multiplier and reduces word storage requirements. If a stem table contains R entries and an affix table has A entries, R x A words can be synthesized, although some of the words will not be meaningful. If an integrated dictionary is formed, the number of words will be $\sum(R_i A_i)$ where R_i is the number of stems that can be associated with A_i specific affixes. In a split dictionary format the affixes can be distributed over all stems and hence the total number of synthesized words will be $(\sum R_i)(\sum A_i)$ and it will always be true that

$$\alpha \ (\sum R_i)(\sum A_i) \geqslant \sum(R_i A_i)$$

where α is the fraction of meaningful words.

Another study [1:63-67] showed that approximately 10 percent of complex word forms contain a prefix whereas 90 percent use suffixes. If a further restriction is placed on the synthesis program that only single-layer word constructions can be made and only suffixes are used, it is estimated that 85 percent of complex words can be accomodated in a split dictionary format.

Thirty-six common suffixes in 43 variant forms (-s, -es; -d, -ed; etc.) were selected for inclusion in the experimental dictionary [1:67-72]. Using single-layer constructions and the 36 suffixes, it is estimated that more than 15,000 words can be synthesized from the split format dictionary, which contains approximately 4,000 word stems.

In searching a dictionary to determine whether an input word is listed or may be synthesized from listed stems and suffixes, it is required to match the input word against the dictionary words. If an equality match is found, the code associated with the word is obtained for transmission. If a match is not found, it must be determined whether the input word can be synthesized.

Decomposition-synthesis routines are based upon the principle of the longest match, where the longest match is the longest stem in a dictionary which is contained in an input word.

Let a_i designate the i-th letter of a word (i = 1, 2, . . . , n) and let s_j designate the j-th letter of a suffix (j = 1, 2, . . . , m). A complex word will then be of the form $a_1a_2 \ldots a_ns_1 \ldots s_m$. If the search for a longest match is successful, a dictionary word, $a_1a_2 \ldots a_n$, will be found such that

$a_1a_2 \ldots a_n \subset a_1^*a_2^* \ldots a_n^*s_1^* \ldots s_m^*$ (read $\subset$ "is contained in"),

the asterisks designating letters of an input word. By subtracting the dictionary word from the input word, the difference, $s_1^* \ldots s_m^*$, is obtained.

A table look-up determines whether or not this is a listed suffix.

The longest match routine will find a listed stem provided that the stem is regular and undergoes no change under inflection or derivation. For stems which undergo change in the process of word formation, special routines must be considered. For example, although _move_ may be listed in the dictionary and _-ing_, in the suffix table, the input word, _moving_, will not be successfully decomposed, for _mov_ is not listed as a stem.

By employing four simple grammatical rules, most of the complex forms of irregular words can be decomposed and synthesized [1:84-91] . These rules are incorporated in the computer routine by assigning a grammatical tag to every dictionary word. The four tags currently employed are:

- _SYNTAG 0_: word can not appear in complex form; irregular form appears as unique word; or word is regular and suffix is added without change in word;
- _SYNTAG 1_: final _E_ of word is deleted upon adding a suffix beginning with a vowel;
- _SYNTAG 2_: final consonant of a word is doubled upon adding a suffix beginning with a vowel;
- _SYNTAG 3_: final _Y_ of word is changed to _I_ upon adding a suffix beginning with letter other than _I_.

By associating a SYNTAG with each dictionary word and by identifying the initial letter of each suffix, simple routines for the synthesis and decomposition of complex words have been devised.

Several factors have been utilized to minimize the search for a longest match. If the number of letters in a dictionary word, W_D, is greater than the number of letters in an input word, W_T, then W_D is not contained in W_T. If the minimal length of a longest match is set at n-letters, then only blocks of words containing the initial n-gram of the input word need be searched. Three letters have been determined to be the optimum search parameter (see Section III).

If a word does not appear in the dictionary and it cannot be synthesized, it must be spelled out. For this purpose, the letters are also encoded. It has been found less costly to encode the letters separately and have two modes of operation: word and letter.

An important consideration in the choice of dictionary format is the coding efficiency that is obtainable from the two formats. Since minimum cost encodings are those that yield the most probable distribution of symbols, encoding of the integrated dictionary meets the conditions for the most efficient coding. It follows, then, that no other coding can be more efficient.

A compromise between storage cost, coding cost, and processing cost points to a modified split dictionary format as the most desirable. In this format, the most frequently used words are incorporated in the dictionary in either canonical or complex form. Since the most frequently occurring words will be included in the modified split dictionary format, the dictionary will also contain the common words that can be synthesized from bound forms and affixes. A cut-off frequency is established to keep the number of words within the limits of the computer storage. Words not in the dictionary can either be synthesized or spelled out while at the same time the efficiency achieved by matching the encoding to the stored dictionary is retained.

It is the modified split dictionary format containing free forms with single layer suffix affixation capability that is used in the experimental system. It should be noted that the modified split dictionary format in actuality makes possible multi-layered word constructions. For example, the presence of the complex forms objection and objective in the experimental dictionary in addition to the free form object makes possible the synthesis of the multi-layered forms objection-able and objective-ly.

Additional considerations concerning the dictionary format from the point of view of computer programming, processing rates, and economic factors are contained in Sections III and VII.

An experimental system embodying the principles previously outlined has been designed and computer programs have been written to test the model system. The operation of the system is shown in the simplified flow charts of Figures 1 - 4. Figure 1 is the encoding routine and shows the preparation of the dictionary tapes. Figure 2 shows the routine for dictionary growth and revision. The heart of the system is contained in the two basic routines of Figure 3, message enciphering at the transmitter and Figure 4, message deciphering at the receiver.

System operation starts with the routine of Figure 1 if the a priori statistics of the information source are known, as in the experimental system. If these statistics are not known, the message enciphering routine of Figure 3 is initially employed. The "new word" tape generated by this routine allows the system to grow its own dictionary for by processing the "new word" tape through the routine of Figure 2, the a posteriori statistics are computed. The frequency sorted words of Figure 2 can be used to generate the initial dictionary encoding or to revise the original encoding in

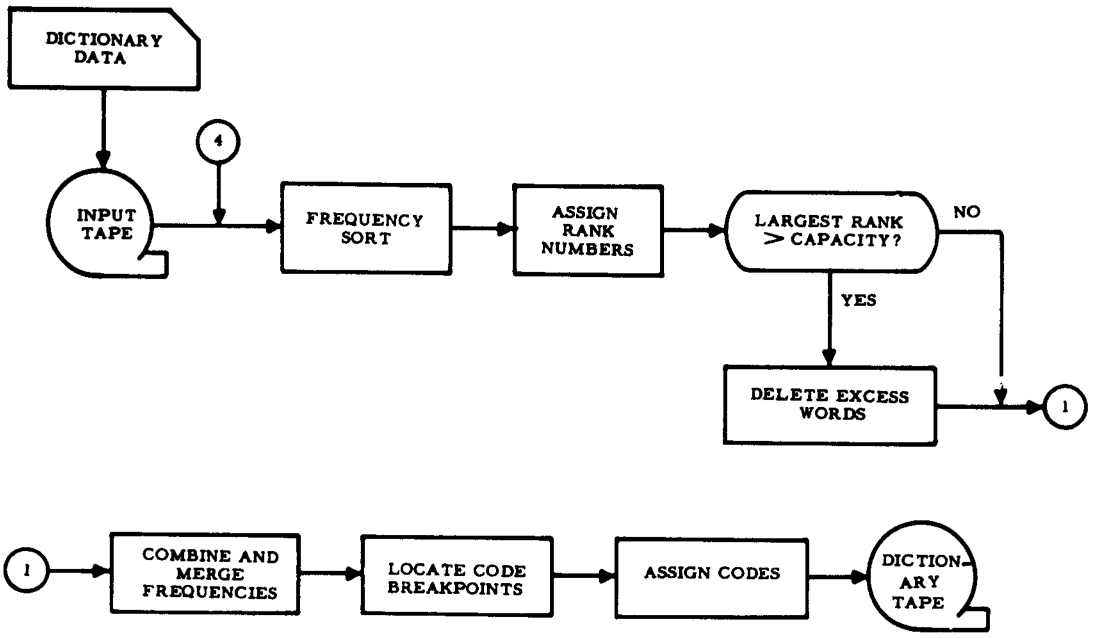


FIG. 1 ENCODING ROUTINE AND PREPARATION OF DICTIONARY TAPES

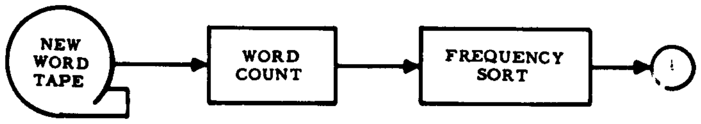


FIG. 2 DICTIONARY GROWTH AND REVISION

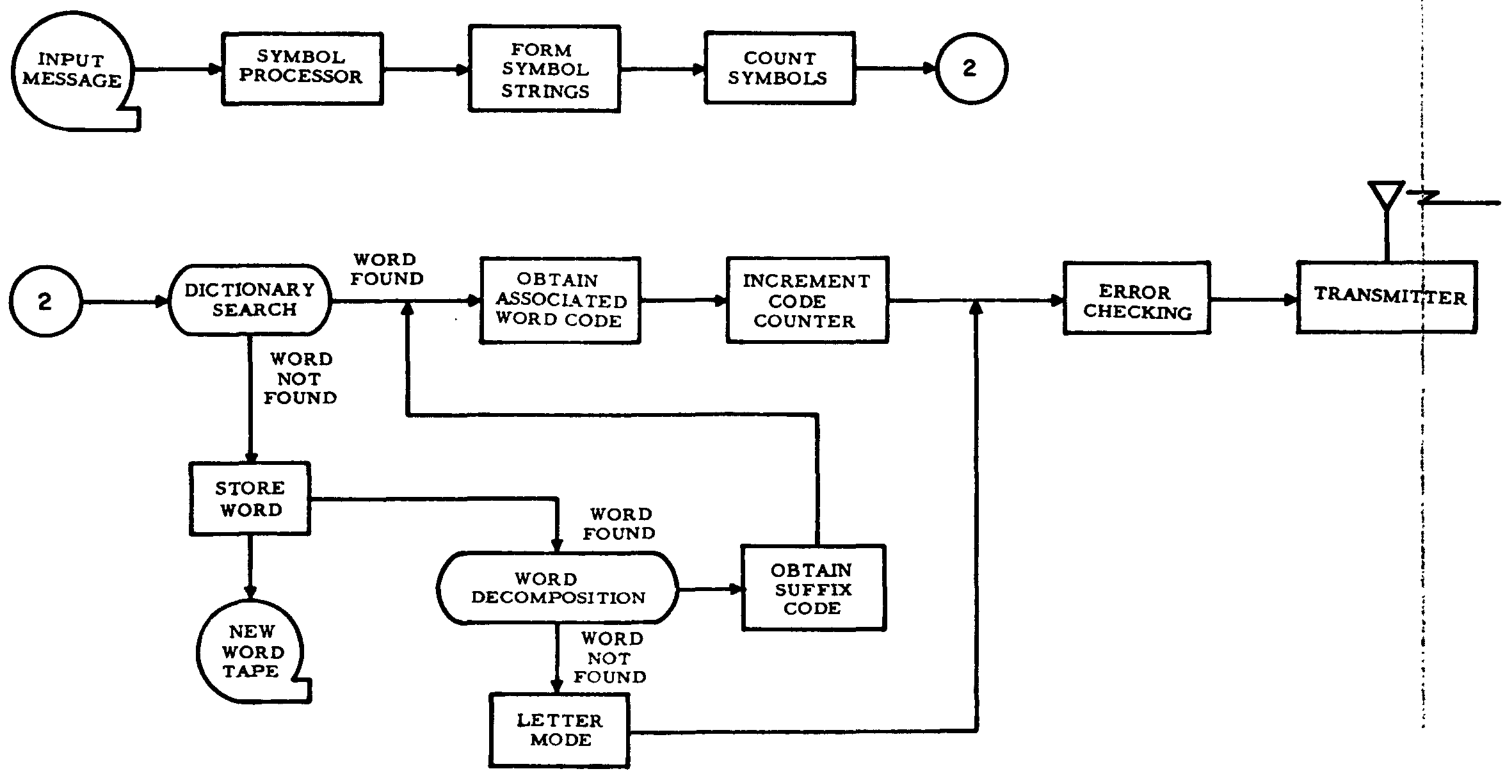


FIG. 3 MESSAGE ENCIPHERING AT TRANSMITTER.

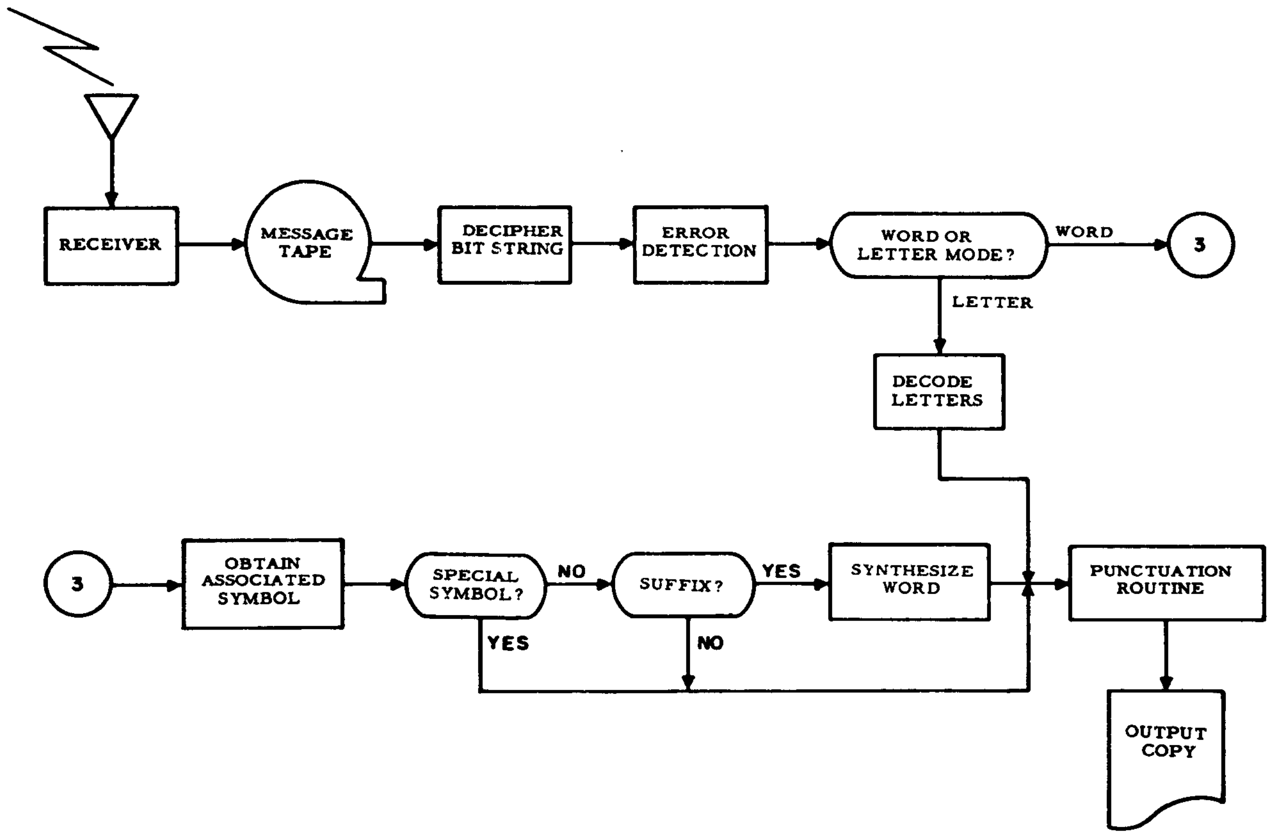


FIG. 4 MESSAGE DECIPHERING AT RECEIVER

conformance with actual usage.

Figures 3 and 4 include the insertion of error checking digits at the transmitter and error detection at the receiver. A detailed description of the error detection system is contained in Section VIII.

Input messages are processed by a computer program which consists of a symbol processor, dictionary search routines, and the encipher routine. Three tables are used to determine the code of a symbol string formed by the character processor: alphabetically sorted dictionary, suffixes, and special symbols (numerals, punctuation). A listing of the beginning of the dictionary is shown in Figure 5 designated TRANSMITTER DICTIONARY.

Each dictionary entry is contained in four computer words. Two computer words contain up to 12 characters of the dictionary word, the numeral, or the special symbol. The third computer word contains the assigned code. The fourth computer word contains the number of characters in the word (or symbol), the SYNTAG designating a grammatical classification, and a counter to record a cumulative frequency count.

In the TRANSMITTER DICTIONARY sample, the code is expressed in octal form, the first binary one on the left indicating the start of the code. The code for the word A, for example, in binary digits is 001110. The first two octal digits at the left of the third column in the table designate the SYNTAG; the two octal digits at the right indicate the number of characters in the word.

Following character processing, word decomposition, and table lookup, a message is enciphered as a concatenation of codes. A sample enciphering is shown under the heading ENCIPHERED MESSAGE in Figure 5.

At the receiver, a single table containing all dictionary entries is stored. This table, a sample of which is shown in RECEIVER DICTIONARY, is arranged in increasing order of the binary codes. By use of a breakpoint table, the number of binary digits in a code is determined, the address of the word represented by the code within the storage area containing that number of digits is calculated, and the word is then located. A print routine synthesizes paradigmatic forms and introduces spaces between words and punctuation symbols as required.

The print out of the message for which the initial binary digits are shown under ENCIPHERED MESSAGE is shown under DECIPHERED and DECODED MESSAGE. Synthesis is denoted by a circumflex. WORDS, for example, was decomposed by the decoder into WORD+S and the codes for WORD and the suffix S were separately transmitted. The receiver recognizes that S is a suffix by a SYNTAG 4 and therefore appends the S to the preceding word without inserting a space between. SKIPPING was enciphered as SKIP+ING, the SYNTAG 2 associated with SKIP instructing the receiver computer to double the final consonant since the suffix ING has a SYNTAG 6 designating a suffix beginning with I. The SYNTAG 14 associated with the period introduces two successive spaces while SYNTAG 10 results in a single space following the comma.

Detailed flow charts and descriptions of the computer programs used to implement the model system are contain in Section IX.

TRANSMITTER DICTIONARY

A	116	0001
ABANDON	174443	0007
ABILITY	75035	0307
ABLE	16402	0104
ABOARD	175755	0006
ABOLISH	777716	0007
ABOUT	1435	0005
ABOVE	16373	0005
ABROAD	174202	0006
ABRUPTLY	75660	0010

* * *

RECEIVER DICTIONARY

,	20	1001
.	21	1401
THE	22	0003
AND	46	0003
A	116	0001
TO	117	0002
OF	120	0002
I	121	0001
IN	122	0002
WAS	123	0003

* * *

ENCIPHERED MESSAGE

10011010101000011000101101111000110001011111000000110111000100110010110101010011011111010101 11

10000011011110001010110101010011100011000000101110110110010001100111110011011000011011111111100

* * *

DECIPHERED AND DECODED MESSAGE

JUST WHAT IS SPEED READING. AND IS IT REALLY READING. IS IT POSSIBLE FOR ANYONE TO SEE AND COMPREHEND THOUSANDS OF WORDS A MINUTE. NOT ALL RAPID READING IS IN ALL REALITY READING. THE AVERAGE READER, SKIPPING NOTHING, MOVES ALONG AT 200 WORDS A MINUTE. EXPERTS AGREE THAT IT IS POSSIBLE TO LITERALLY READ NEARLY 500 WORDS A MINUTE, PERHAPS MORE, BUT NOT, THEY INDICATE, THOUSANDS. THE EYE SIMPLY CAN NOT SEE THOUSANDS OF WORDS A MINUTE. AT SUCH RATES, ONE CAN SKIM, PICK UP IMPORTANT PHRASES, GET THE ESSENTIALS OF THE MATERIAL, BUT CERTAINLY NOT SEE EVERY WORD. #

FIGURE 5. MESSAGE PROCESSING THROUGH TRANSMITTER AND RECEIVER.

III. EXPERIMENTAL DICTIONARY

A language can be considered as an infinite set of words since new words can always be created. By limiting the word set to those words listed in a standard dictionary, a finite set can be obtained although the word types included in the dictionary may range as high as 600,000. It is apparent that a smaller subset of the language must be selected in order to remain within the storage capacities of a computer, minimize word search time, and meet the objective of efficient transmission of enciphered text drawn from the universe of words.

Table 1 is a summary of several word counts that have been made and shows the number of word types and word tokens in the various samples.

TABLE 1

WORD COUNTS

STUDY		DATE	MATERIAL	WORDS		%
				TOKENS	TYPES	TYPES
Eldridge	[2]	1911	Newspaper articles	43 989	6,002	13.6
Dewey	[3]	1923	Miscellaneous	100,000	10,161	10.2
Hanley	[4]	1937	Joyce's "Ulysses"	260,430	29,899	11.5
Thorndike	[5]	1944	Miscellaneous	18,000,000	30,000*	-
Miller-Newman	[6]	1958	Miscellaneous	36,299	5,537	15.2
Armour Res.	[1]	1960	Military exercise	38,992	2,081	5.3

* The 30,000 listed words are those that occur with a frequency of at least 4 per 18 x 10^6 words. The total number of word types is not indicated. Regular plurals, comparatives and superlatives, inflected forms and past participles ending in n were counted under the stem word.

A word type is one that differs in one or more symbols from every other word in a dictionary. Thus, count and counts are two word types. Count as a title and count as a number determined by counting, or count as a noun and count as a verb are considered one word type. The summation of the frequencies of word types is the aggregate frequency of word tokens.

It is noted that the percentage of word types varies between approximately 10 and 15 for all samples but the military exercise. The 5.3 percent of this sample reflects the high redundancy in a specialized vocabulary.

Eldridge found that 250 word types include most of the chief sentence forming words of the English language, while 750 word types constituted more than 75 percent of his aggregate sample. Dewey found that 9 word types formed over 25 percent of his sample, 69 words formed 50 percent, and 732 words formed over 75 percent. The "Ulysses" count showed that 1 000 word types made up 80 percent of the aggregate. In the military exercise, 25 word types comprised 51. 7 percent of the aggregate, 50 words 61. 7 percent, 100 words 72. 9, and 500 words 91. 5 percent.

On the basis of these studies, it is indicated that a vocabulary of between 500 and 1,000 word types can constitute the basis of a dictionary which will cover approximately 75 percent of any word sample. Beyond these basic words, it appears that no statement of the inherently most common words is possible, independent of subject matter.

It is this consideration, in addition to the numerous problems entailed in selecting representative text and making an extensive word frequency count, that makes the capability of a system to grow its own dictionary and conduct its own census not only an attractive feature but also a necessary one.

The dictionary word list has been obtained from Thorndike and Lorge "The Teacher's Word Book of 30,000 Words" [5]. The book summarizes a count of 18,000,000 words obtained from four different samples, each of approximately 4,500,000 words. The four samples might be considered: 1) classical, 2) popular magazines, 3) juvenile literature, 4) miscellaneous.

The 19,440 words occurring once or more per million are listed alphabetically, showing the frequency counts in each of the four samples and the average frequency per million words from the 18,000,000 word total count. An additional 9,202 words that occurred less than once per million but oftener than four per 18,000,000 are also listed alphabetically with average frequencies indicated. An additional 1,358 words that occurred four times per 18,000,000 words are listed, for a total of 30,000 words. The list contains proper nouns, and contractions.

In most cases, distinction was not made between members of a paradigmatic set. Thus regular plurals, comparative and superlatives, verb forms ending in s, ed, and ing, past participles formed by adding n, and several rare forms are counted in together with the stem. In general, participial adjectives and verbal substantives were also included under the stem.

The frequencies listed in sample 2, the Lorge magazine count, were used to select the experimental dictionary since this count was the most complete and its source (five popular magazines circa 1927-1938) appeared to be most representative of currently used English.

A cutoff frequency of 50 was chosen in order to obtain approximately 5,000 of the most frequent words. Archaic words (knight, behold, etc.) and other words not in general usage (specialized foods in recipes, etc.) were not considered regardless of their frequencies.

Frequencies for the split dictionary format were obtained by adding the frequencies of derivational forms (< 50) to the stem as illustrated below.

experiment	244	(-s, -ed, -ing included)
experiment -al	40	
experiment -al -ly	5	
experiment -ation	14	
experiment -er	3	
	306	

Thus the word experiment appears in the dictionary with a frequency of 306.

When members of a paradigmatic set included complex forms with a frequency equal to or greater than 50, they were listed as word types, while the complex words with frequencies less than 50 were combined with the stem or complex word to which a suffix was joined as shown below.

direct	416	+ ness - 14	430
direct -ion			441
direct -ly			258
direct -or	315	+ y - 21	336

The number of word types thus selected totalled 5,153. The word length distribution of these words is given in Table 2.

TABLE 2

WORD LENGTH OF DICTIONARY WORDS

LETTERS	NO. WORD TYPES
1	2
2	32
3	275
4	839
5	936
6	895
7	771
8	527
9	362
10	260
11	137
12	79
13	30
14	8
TOTAL	5,153

The average word length is 6.35 letters per word type and the weighted average of 4,257,692 word tokens is 3.95 letters per word. The weighted average is slightly less than the often cited value of 4.5 letters per word. The word length distribution is shown in Figure 6.

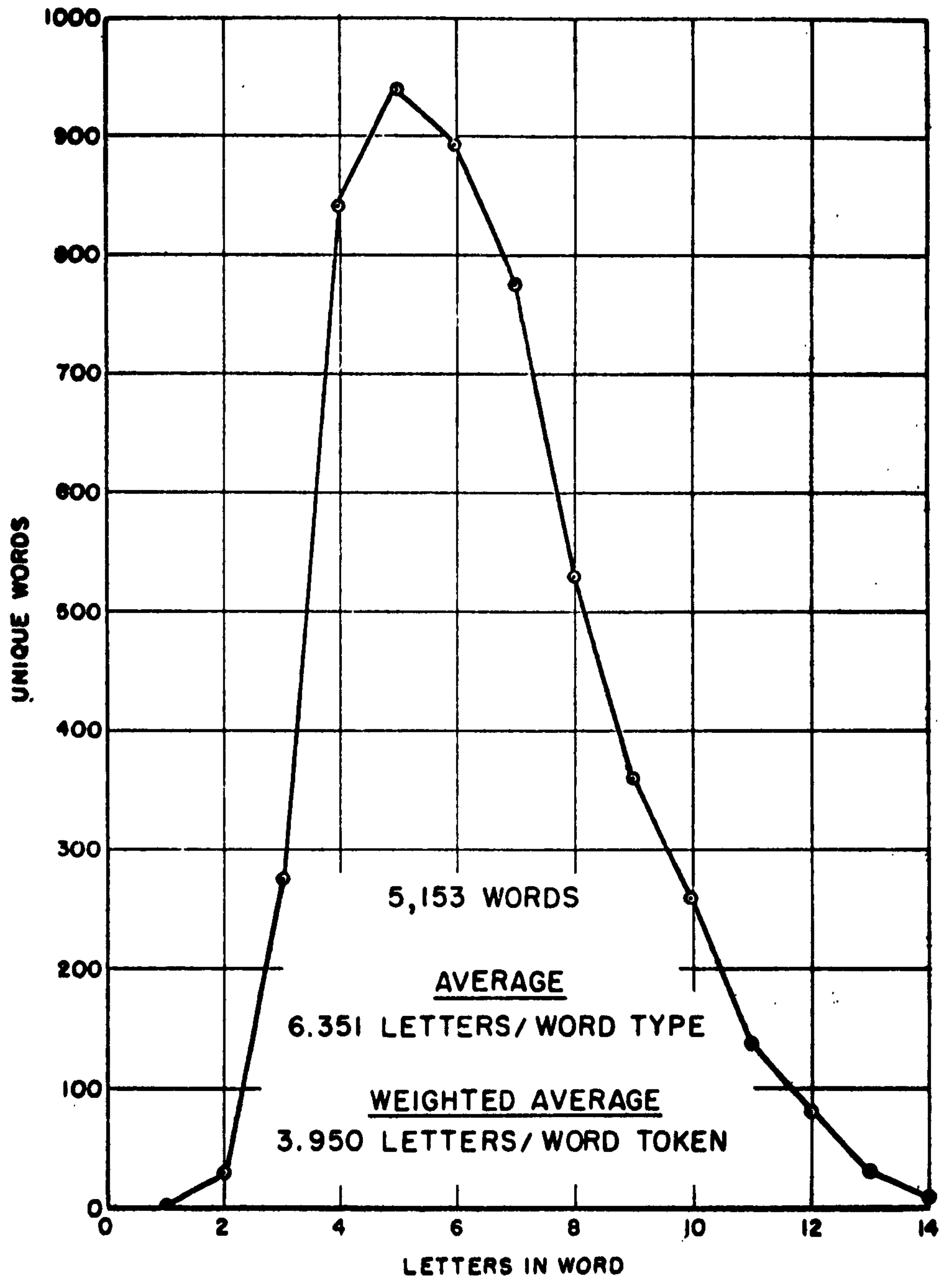


FIG. 6 WORD LENGTH OF EXPERIMENTAL DICTIONARY

It is noted in Table 2 that only 38 of the 5,153 listed words are greater than 12 letters in length. Of these 38 words, 22 have frequencies less than 100, 13 have frequencies between 100 and 199, and only three have frequencies greater than 200. The highest frequency of the latter group is 338. All but eight of the 38 words can be synthesized in the split dictionary, using the limited number of suffixes provided. Hence it is possible, without great loss of flexibility, to simplify computer storage (in a fixed word length computer) by eliminating these words from a dictionary and establishing a maximum word length of 12 letters.

In general, it is found that nearly all words greater than 12 letters in length occur infrequently, are complex, and therefore can be synthesized [1:78-83] .

The selection of suffixes has been discussed in detail in [1:67-72] . Of the 90 most common suffixes studied by Thorndike [7] , 36 with adjusted frequencies greater than 50 were selected for inclusion in the experimental dictionary. A count was made of their frequency of occurrence in the first 19,440 words listed in Thorndike's Word Book. A suffix was counted 1) if it appeared in a word with a frequency equal to or greater than 50, 2) if the canonical form of a word had a frequency of 50 or greater, 3) if the sum of the paradigmatic set of a stem had a frequency of 50 or greater. The suffix count is listed in Table 3.

Since a modified split dictionary format was selected for the experimental dictionary, and complex words with a frequency equal to or greater than 50 were listed as word types, the suffix count was modified accordingly. The adjusted count is shown in the last column of Table 3.

TABLE 3

SUFFIX FREQUENCIES IN EXPERIMENTAL DICTIONARY

NO.	SUFFIX	UNIQUE WORDS	AGGREGATE FREQUENCY	ADJUSTED FREQUENCY*
1	-ly	699	51, 124	8, 102
2	-er	390	23, 506	4, 000x
3	-y	240	19, 138	3, 301
4	-ion	216	18, 379	2, 364
5	-en	101	12, 752	700
6	-al	169	10, 108	2, 224
7	-ness	167	7, 991	1, 813
8	-ment	87	7, 966	823
9	-ful	143	7, 705	2, 050
10	-ation	95	5, 072	1, 245
11	-ity	92	4, 203	1, 143
12	-or	83	3, 981	930
13	-ive	87	3, 394	1, 109
14	-able	92	3, 312	1, 086
15	-ous	67	3, 061	878
16	-ize	44	3, 022	416
17	-ant	37	2, 948	288
18	-age	37	2, 825	362
19	-less	108	2, 696	1, 324
20	-ance	38	2, 607	454
21	-ent	34	2, 453	454
22	-ence	23	1, 995	342
23	-ial	28	1, 372	369
24	-ship	31	1, 356	375
25	-ic	53	1, 339	711
26	-ish	24	1, 089	239
27	-ure	10	1, 070	38+
28	-ate	21	965	199
29	-th	15	962	191
30	-ary	29	869	318
31	-ist	45	695	442
32	-hood	12	585	172
33	-ism	24	408	298
34	-d	?	?	1, 500x
35	-ed	?	?	10, 000x
36	-n	?	?	300x
37	-nce	?	?	100x
38	-nt	?	?	100x
39	-r	?	?	638x
40	-st	?	?	500x
41	-est	?	?	1, 500x
42	-s	?	?	10, 000x
43	-es	?	?	3, 000x
44	-ing	?	?	10, 000x

* Suffix frequency after unique words with frequencies equal to or greater than 50 were entered into dictionary

+ Deleted from list because frequency is less than 50

x Estimated frequency

Frequencies for the plural endings and verb inflections were estimated, mainly by extrapolating the results of Dewey's word count. It is noted in Table 3 that seven suffixes that begin with _e_ appear in two forms:

-_d_, -_ed_	-_nt_, -_ent_
-_n_, -_en_	-_r_, -_er_
-_nce_, -_ence_	-_st_, -_est_
-_s_, -_es_	

These double entries were made in the computer suffix table to simplify the word decomposition and synthesis routines. The division of the frequency count between the two forms was also estimated.

The 5, 153 word types in the dictionary include 3, 894 stems (free forms), 1, 094 complex forms whose stems are independently listed, and 165 complex forms whose stems are not listed. From the 3, 894 stems and the 36 listed suffixes it is possible to synthesize an additional 2, 261 word types that are included among the 19, 440 most frequent words listed by Thorndike. This is exclusive of plurals, verb inflections, and degrees.

The synthesis tags (SYNTAG) are divided as follows:

SYNTAG 0		3, 819
SYNTAG 1	819	
SYNTAG 2	212	
SYNTAG 3	303	1, 334

For experimental purposes, 27 geographical names are listed in the dictionary together with the ten numerals and 14 punctuation and special symbols. Eliminating the 38 words greater than 12 letters in length as well as the word _mister_ since its abbreviation _mr_ appears, a total of 5, 208 entries are included.

Two listings of the experimental dictionary appear in Appendices A and B of [1]. The first listing is an alphabetical arrangement of 5,153 words. The number of letters in the word, its frequency (in 4.5×10^6 words of magazine text), the Thorndike rank (average frequency per million words from a total of 18×10^6 words), the synthesis tag (SYNTAG), identification as to stem or complex form, and whether it is one of the 850 Basic English words, are indicated for each word. A list of numerals, punctuation, and special symbols is also included, together with a list of suffixes and geographical names. A list of contractions is included for reference, although these were not used in the dictionary. The second listing, Appendix B, is ordered by frequency. The rank number (ties ignored), the word, and its frequency are listed in ascending rank.

The frequency distribution of the 5,153 most frequent words is shown in Table 4.

TABLE 4

FREQUENCY DISTRIBUTION 5,153 MOST FREQUENT WORDS

FREQUENCY	NUMBER WORD TYPES	CUMULATIVE NUMBER
> 100,000	5	5
10,000 - 99,999	55	60
1,000 - 9,999	434	494
900 - 999	64	558
800 - 899	68	626
700 - 799	83	709
600 - 699	129	838
500 - 599	144	982
400 - 499	212	1,194
300 - 399	371	1,565
200 - 299	586	2,151
100 - 199	1,361	3,512
50 99	1,641	5,153

The 494 most frequent words (frequencies $\geq$ 1000) comprise 76.2 percent of the tokens in the dictionary, a figure which is consistent with the earlier studies cited in Table 1.

The log-log plot of word frequency versus word rank in Figure 7 exhibits the familiar Zipf-Mandelbrot characteristics.

The distribution of word types and word tokens by the first letter of the dictionary words is of interest in programming routines for word search and word decomposition. Table 5 lists the letters of the alphabet with the number of word types and word tokens found for each letter together with an associated rank according to frequency.

The letter S is the most frequent initial letter among word types, appearing in 664 dictionary words. C ranks second with 501 types and P is third with 402 types. In the fourth rank is A with 321 types and D appears fifth with 310 types.

Among word tokens, the rank order shows little correlation with the type ranks. Thus T is the initial letter of word tokens with the greatest frequency although in word types, T ranks sixth. The three words, the, to, and that, account for 63 percent of the word tokens beginning with T. A ranks second as the initial letter in tokens, 50 percent of the number of tokens being represented by the two words a and and. I ranks third in tokens, nearly 50 percent of the total tokens appearing in the two words I and in. W and H rank fourth and fifth, respectively, in the token list.

If the search for a longest match starts with the first letter of an input word, then by Table 5, from 664 (S) to 2 (Z) words must be searched. Making the initial n-gram two letters, reduces the maximum number of searches to 224 (CO) and a minimum of one (ZE). By making the minimum

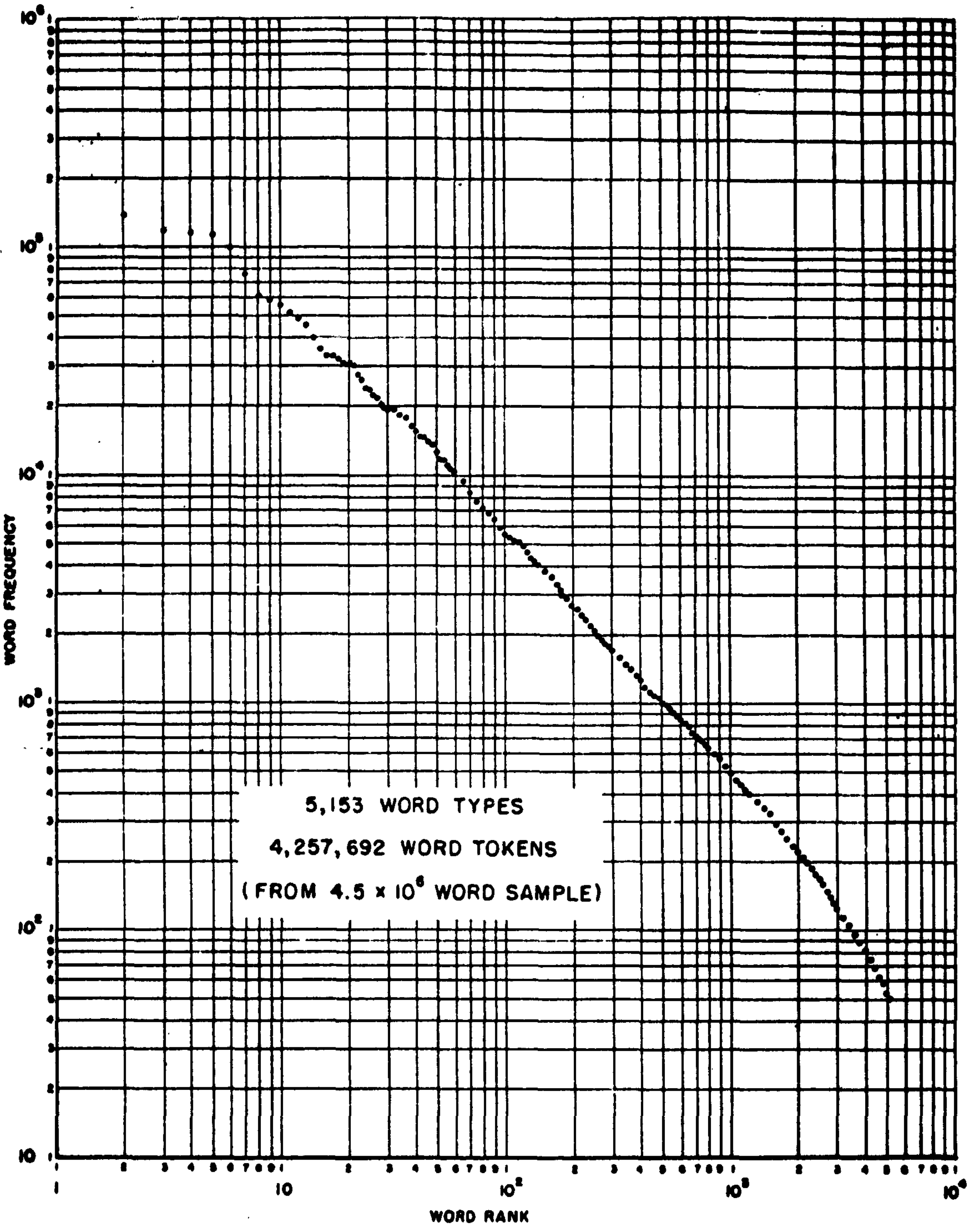


FIG. 7 RANK-FREQUENCY CHARACTERISTICS OF 5,153 MOST FREQUENT WORDS

TABLE 5

RANK OF TYPE/TOKEN WORDS IN DICTIONARY BY

FIRST LETTER OF WORD

LETTER	WORD TYPES	TYPE RANK	TOKEN RANK	WORD TOKENS
A	321	4	2	515,656
B	286	7	9	175,809
C	501	2	10	166,742
D	310	5	13	119,835
E	226	11	18	78,274
F	266	9	11	166,518
G	159	16	16	83,432
H	198	12	5	295,335
I	191	13	3	343,210
J	40	21	23	18,192
K	31	22	21	24,518
L	179	14.5	14	116,924
M	235	10	8	188,566
N	95	18	15	90,019
O	117	17	7	263,240
P	402	3	12	123,334
Q	22	23	24	8,487
R	274	8	17	80,060
S	664	1	6	288,978
T	289	6	1	647,530
U	73	20	20	42,916
V	74	19	22	18,209
W	179	14.5	4	325,421
X	0	26	26	0
Y	19	24	19	76,386
Z	2	25	25	101
WORD TOTALS	5,153			4,257,692

length of the longest search three letters, the search time is further reduced as shown in Table 6 which lists the 68 most frequent initial trigrams of the 5, 153 word dictionary. The maximum number of words searched for a longest match of the initial trigram is 96 (CON) with other frequencies as shown in Table 6 and ranging down to one.

Table 7 lists the distribution of initial trigrams with first letter S, the first letter of the greatest number of word types in the dictionary. Among the 664 words beginning with S, there are 120 initial trigram sets ranging from 1 to 38 members; the average number of words per trigram set is 5. 53. Among the 501 words beginning with C, the initial letter of the second largest set of word types, there are 64 trigram sets ranging from 1 to 96 members, the average being 7. 82 words per initial trigram. For initial letter P, there are 75 trigram sets ranging from 1 to 60 members with an average of 5. 36 words per initial trigram.

Since setting the initial n-gram equal to four would eliminate from the synthesis program all 2- and 3-letter words plus all 4-letter words of SYNTAG 1 (final E deleted), 3 was chosen as the best value of n. Two-letter and 3-letter words of SYNTAG 1, accordingly, must be classed under SYNTAG 0 and complex words obtainable from these stems must be listed as word types. Inasmuch as there are eight 3-letter stems of SYNTAG 1 and nine 2-letter stems from which complex words can be obtained, only 17 stems are affected by establishing the initial trigram as the minimal length word for a longest match.

When a word search fails to find an equality match, the search will terminate at a point where the numerical value of the next dictionary word is greater than that of the input word, and the numerical value of the

TABLE 6

INITIAL TRIGRAMS

1.	CON	96	24.	IMP	19	47.	CHE	13
2.	PRO	60	25.	REC	19	48.	COL	13
3.	COM	51	26.	IND	18	49.	DEL	13
4.	DIS	47	27.	PRI	18	50.	EVE	13
5.	STR	38	28.	SHA	18	51.	MAN	13
6.	RES	33	29.	SHO	18	52.	REF	13
7.	FOR	31	30.	THR	18	53.	TRI	13
8.	PRE	30	31.	CAR	17	54.	ELE	12
9.	STA	30	32.	LEA	17	55.	HAR	12
10.	GRA	27	33.	REA	17	56.	MAR	12
11.	INS	27	34.	THE	17	57.	MIN	12
12.	PER	27	35.	WOR	17	58.	MIS	12
13.	CHA	25	36.	DEC	16	59.	OUT	12
14.	TRA	25	37.	STE	16	60.	PAT	12
15.	INT	24	38.	STO	16	61.	POS	12
16.	EXP	23	39.	ATT	15	62.	REP	12
17.	DES	22	40.	INV	15	63.	SEN	12
18.	APP	21	41.	PLA	15	64.	SUP	12
19.	SPE	21	42.	REL	15	65.	THI	12
20.	ACC	20	43.	ASS	14	66.	TRE	12
21.	HEA	20	44.	STU	14	67.	WAR	12
22.	PAR	20	45.	SUR	14	68.	WEA	12
23.	COU	19	46.	BRI	13	All Others: 1-11		

TABLE 7

INITIAL TRIGRAMS - FIRST LETTER S

SAC	3	SCR	9	SHY	1	SLO	3	SOR	5	SUC	6
SAD	3	SEA	6	SIC	3	SLU	2	SOU	8	SUD	2
SAF	3	SEC	7	SID	2	SMA	3	SOW	1	SUE	1
SAG	2	SEE	7	SIF	1	SME	1	SPA	4	SUF	4
SAI	4	SEI	1	SIG	6	SMI	1	SPE	21	SUG	3
SAK	1	SEL	8	SIL	7	SMO	4	SPI	9	SUI	5
SAL	10	SEN	12	SIM	6	SNA	4	SPL	4	SUL	2
SAM	2	SEP	3	SIN	10	SNE	2	SPO	7	SUM	3
SAN	5	SER	9	SIR	3	SNI	1	SPR	7	SUN	8
SAS	1	SES	1	SIS	1	SNO	2	SPU	1	SUP	12
SAT	6	SET	3	SIT	3	SOA	2	SPY	1	SUR	14
SAU	1	SEV	7	SIX	4	SOB	3	SQU	4	SUS	6
SAV	3	SEW	1	SIZ	1	SOC	3	STA	30	SWA	6
SAW	1	SEX	1	SKA	1	SOD	1	STE	16	SWE	11
SAY	2	SHA	18	SKE	1	SOF	4	STI	10	SWI	5
SCA	10	SHE	11	SKI	9	SOI	1	STO	16	SWO	3
SCE	3	SHI	9	SKY	1	SOL	11	STR	38	SWU	1
SCH	4	SHO	18	SLA	6	SOM	9	STU	14	SYM	4
SCI	3	SHR	7	SLE	6	SON	2	STY	1	SYN	1
SCO	6	SHU	4	SLI	9	SOO	2	SUB	11	SYS	1

preceding dictionary word is less than that of the input word. This will be true because the alphabetic ordering of the dictionary implies numerical ordering since the BCD (Binary Coded Decimal) code is an increasing function of the alphabetic order. The variable length dictionary words will not affect the procedure if the words are left-justified, i.e., the first character of the dictionary word occupies the left-hand first six bits of the computer word. The information stored in this way can be regarded as a 36 bit binary fraction with the decimal point at the left of the 36th bit. Thus, attic will be numerically less than cat. If right justification were used, cat would be numerically less than attic.

The direction of search for the longest match may be determined by analysis of the SYNTAG classes as shown in Table 8. The a_i designate letters of a dictionary word, the a_i^* designate letters of an input word, and the s_i^* designate letters of a suffix.

From Table 8 it is seen that the search for a longest match by word decomposition must be bidirectional, starting from the point where the equality search terminated. The boundaries for the search are the equality match of the initial trigram.

Since 74 percent of the SYNTAG's of dictionary words are O, the numerical value of the complex form of most of the words that will be encountered in input text will be greater than the dictionary stem. Similarly, the numerical value of complex words of SYNTAG 1 and SYNTAG 2 will be greater than the stem, with the exception of suffixes beginning with a of SYNTAG 1. Hence, the word decomposition routine always proceeds backward from the word where the equality match failed, and continues in the forward direction only if the backward search also fails. A detailed

TABLE 8

NUMERICAL VALUES OF STEM AND COMPLEX FORM IN WORD DECOMPOSITION

SYNTAG	NUMERICAL VALUE		EXAMPLE
SYNTAG 0		$a_1a_2 \ldots a_n < a_1^* a_2^* \ldots a_n^* s_1^* \ldots s_m^*$	work < working
SYNTAG 1	Case 1, $s_1 = \underline{A}$:	$a_1a_2 \ldots a_{n-1}\, \underline{E} > a_1^* a_2^* \ldots a_{n-1}^* s_1^* \ldots s_m^*$	love > lovable
$a_{n_i} = \underline{E}$	Case 2, $s_1 = \underline{E}$:	$a_1a_2 \ldots a_{n-1}\, \underline{E} < a_1^* a_2^* \ldots a_{n-1}^* s_1^* \ldots s_m^*$	love < loving
$s_1 = \underline{A}, \underline{E}, \underline{I}, \underline{O}, \underline{U}, \underline{Y}$	Case 3, $s_1 \neq \underline{A}, \underline{E}$:	$a_1a_2 \ldots a_{n-1}\, \underline{E} < a_1^* a_2^* \ldots a_{n-1}^* s_1^* \ldots s_m^*$	love < lover
SYNTAG 2			
a_{n_i} = consonant		$a_1a_2 \ldots a_{n_1} < a_1^* a_2^* \ldots a_{n_i}^* a_{n+1_i}^* s_1^* \ldots s_m^*$	hit < hitting
$s_1 = \underline{A}, \underline{E}, \underline{I}, \underline{O}, \underline{U}, \underline{Y}$			
SYNTAG 3			
$a_{n_i} = \underline{Y}$	Case 1, $s_1 = \underline{I}$:	$a_1a_2 \ldots a_{n-1}\, \underline{Y} < a_1^* a_2^* \ldots \underline{Y}\, s_1^* \ldots s_m^*$	reply < replying
$s_1 = \underline{A}, \ldots, \underline{Z}$	Case 2, $s_1 \neq \underline{I}$:	$a_1a_2 \ldots a_{n-1}\, \underline{Y} > a_1^* a_2^* \ldots a_{n-1}^* s_1^* \ldots s_m^*$	reply > replies beauty > beautiful

description of the decomposition routine accompanies Chart 7.6 in Section IX.

Failure of the equality match and the backward-forward search indicates that the input word is neither in the dictionary nor can it be synthesized. The word is spelled out, therefore, letter by letter.

The word synthesis routine at the receiver is the inverse of the word decomposition routine at the transmitter. A decoded word is stored until a second word is decoded. If the second word is not a suffix, the first stored word is printed out. If the second word is a suffix, the synthesis routine is called up. This routine examines the SYNTAG of the first word and the first letter of the suffix. A SYNTAG 0 stem calls for the affixation of the suffix without regard to the first letter of the suffix. A suffix that begins with a consonant calls for similar affixation without regard to the SYNTAG.

The conjunction of SYNTAG 1 and an initial vowel in a suffix, results in the dropping of the terminal <u>E</u> of the stem before affixation. Similarly, conjunction of SYNTAG 2 and an initial suffix vowel, calls for a doubling of the final consonant of the stem before affixation; SYNTAG 3 and an initial letter other than I causes the final <u>Y</u> of the stem to be changed to <u>I</u> before affixation.

A space routine at the receiver output, generates spaces as required for intelligible reading of output copy. Spaces are suppressed between stems and their suffixes and between numerals and other symbols, unless called for. Each punctuation symbol calls up its required space format.

Flow charts and a description of the computer programs for word synthesis and spacing appear in Section IX.

The encoding of the 5,208 dictionary items is discussed in the next section in the context of an investigation into the general characteristics of variable length encodings.

IV. CHARACTERISTICS OF EXHAUSTIVE PREFIX ENCODINGS

A. Introduction

Characteristics of minimum-redundancy encodings have been described in a number of papers [8-10]. The Shannon-Fano encoding is uniquely decipherable and has the prefix property but is not necessarily exhaustive. The Huffman encoding has both the prefix and exhaustive properties and by ensuring that every possible sequence of binary digits is used as a code or as the prefix of a code, the cost is reduced and approaches the minimum theoretical value. Gilbert and Moore [11] and Karp [12] have investigated the properties of exhaustive prefix encodings in more detail. The latter presents an algebraic method for constructing an encoding by use of integer programming as contrasted with what is essentially a graphical solution used by Huffman.

Additional characteristics of the exhaustive prefix encoding (hereafter referred to as EPE) are described in this section. The Huffman encoding is used as a model since it has been shown [11] that this encoding is a minimum cost encoding. A computer program for generating an encoding is described. It is shown that although the average length of an encoding remains constant regardless of the decision rule for merging frequencies, the method of merger does influence the maximum length code of the encoding and the cumulative number of digits in all the codes. A program for generating a canonical encoding is also described.

The efficiency of the encoding for the items of the experimental dictionary is calculated and the structure functions of the canonical encoding are tabulated. The entropy of the dictionary is also calculated.

A proof for the exhaustive property of an encoding is given, the derived relationship serving as a convenient method for checking that an encoding is exhaustive. An investigation is made of the structure of EPE's, the total number of EPE's that can be generated for a given maximum code length, and the distribution of the number of codes contained in these encodings. Some results on the non-uniqueness of optimum solutions for encoding are also presented.

B. Generating an Optimum EPE

Given a set of symbols, $W = \left\{ w_1, w_2, \ldots, w_n \right\}$ and a source generating sequences of these symbols, a code is defined as a finite sequence of channel letters. The channel letters $b_1, b_2, \ldots, b_n$ may be of equal or unequal length. In this section the binary channel with letters 0 and 1 will be assumed. An encoding associates a code C_i with each symbol w_i of the symbol set. A set of codes is an encoding, $\Gamma = \left\{ C_1, C_2, \ldots, C_n \right\}$. An enciphering is a concatenation of codes selected from Γ in which each C_i is substituted for its corresponding source symbol. Each symbol and hence each code appears with probability p_i.

An encoding is uniquely decipherable if, for each finite enciphered message, there exists no other concatenation of codes which could produce this enciphering. An encoding has the prefix property if no code is a prefix of any other code of the encoding. The encoding is exhaustive if it encodes two or more symbols in a uniquely decipherable manner and, for every infinite sequence of channel letters, there is some message which can be enciphered as that sequence.

An optimum EPE is that minimum redundancy encoding which minimizes the average number of channel letters in a message ensemble consisting of a finite number of messages. Mathematically it is

$$\min L_{av} = \sum_{i=1}^{n} p_i L_i$$

where L_i is the length of the code of symbol w_i and p_i is its relative frequency.

Huffman's graphical method for finding an optimum encoding is based on the construction of a frequency tree. The two lowest frequencies which comprise a prefix set are first combined and the combined frequency is merged in the diminished list of frequencies. The next two lowest frequencies are then combined, the resulting frequency again being merged in the diminished list. Given N frequencies, the procedure is terminated after N-1 combinations. A detailed discussion of the Huffman algorithm is given in Gorn [13].

Huffman stated that it is possible to rearrange (merge) combined frequencies in any manner among equally likely frequencies without affecting the average code length [10]. Mandelbrot [14] has also pointed out that encoding procedures can be equal in cost even when their trees are not isomorphic. Gorn in his proof of the Huffman algorithm showed that a permutation of the branches of a tree at the nodes gives rise to a class of trees whose total path lengths are equivalent.

Although the value of $\sum p_i L_i$ remains constant regardless of the decision rule for merging combined frequencies (provided they are merged in such a way that the least two frequencies are always combined) the values of $\sum L_i$ and L_{max}, the length of the longest code have been found to be dependent on the merger rule. Minimization of $\sum L_i$ and especially L_{max} can be significant in computer programming and storage allocation. By adopting different merging rules, a variety of frequency trees can be formed in which $\sum L_i$ and L_{max} vary.

To show that the sum of the lengths of all paths is dependent upon the merging rule and to prove that bottom merging results in an optimum EPE with minimum $\sum L_i$ and minimum L_{max}, the binary tree is introduced.

A tree is a directed graph consisting of a set of points called nodes and a set of ordered pairs of these points called branches. $a_1 a_2$ designates the branch joining node a_1 to node a_2. a_1 is the initial node or boundary and a_2 is the terminating node or boundary of the branch. A collection of branches of the form a_1a_2, a_2a_3, ..., $a_{n-1}a_n$ is a path. The path is called a circuit if the initial and terminating nodes of a path are identical.

The degree of a node of a directed graph is the number of branches of the graph for which it is a boundary. The out-degree of a node of a directed graph is defined as the number of directed branches from that node; the in-degree of a node of a directed graph is defined as the number of directed branches to that node.

A binary tree is now defined as a directed graph which contains no circuits and has the following properties [15] :

i. There is one and only one node, called the root, such that for any node, a, there exists one and only one path which begins with the root and ends with a.

ii. The in-degree of each node, a, is either two or zero.

It is obvious that the out-degree of each node is one with the exception of the root whose out-degree is zero.

The node is said to be a proper node if the in-degree is two and a blank node (leaf) if the in-degree is zero. A proper node is a pure proper node if it is the terminating boundary of two branches whose initial boundaries are both proper nodes. It is a regular proper node if it is the terminating boundary of two branches whose initial boundaries are both blank nodes. A proper node is a mixed proper node if it is the terminating boundary of two branches whose initial boundaries are one blank node and one proper node.

The length of a path is the number of proper nodes in a path, that is, the number of nodes in a path from a blank node to and including the root.

A level k of a tree consists of those nodes which lie on all paths of length k from a root. The root lies on the first level. It follows that the number of possible nodes on the k-th level of a binary tree is 2^{k-1}. A deficient level is defined as a level k which contains less than 2^{k-1} proper nodes. The number of missing nodes is called the deficiency of the level.

Let us now consider a binary tree with i blank nodes. We will first determine the least upper bound and the greatest lower bound of the sum of the lengths of all paths, each path beginning with a different blank node and terminating in the root. That such bounds exist is evident from the following:

1. Least upper bound

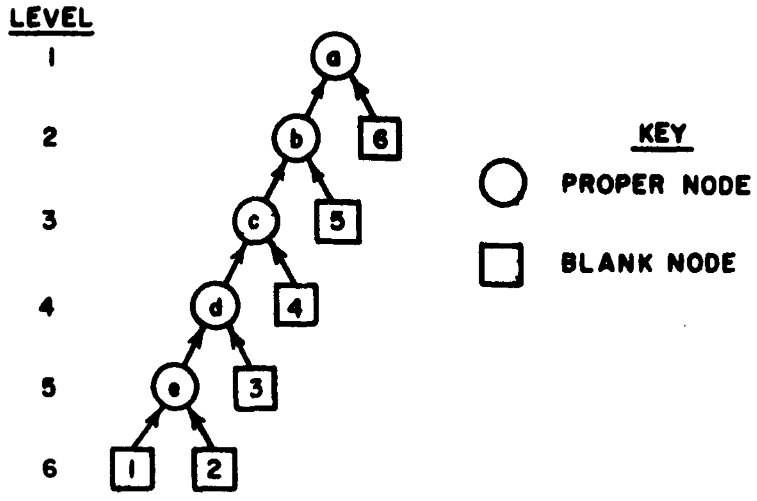
A binary tree with i blank nodes has exactly i-1 proper nodes. The sum of the lengths of all paths $\sum L_i$ can be maximized by arranging the proper nodes of the tree so that exactly one proper node appears on each level. This is illustrated in Figure 8A for the case of i = 6. $L_6' = 5 + 5 + 4 + 3 + 2 + 1 = 20$ is maximum and hence L' is the least upper bound of the sum $\sum L_i$.

The maximum number of levels v (including a level containing only blank nodes) is clearly v = i and the max $\sum L_i$ is the arithmetic progression $\max \sum L_i = (2 + 3 + \ldots + i) = 1/2\,(i - 1)\,(i + 2)$.

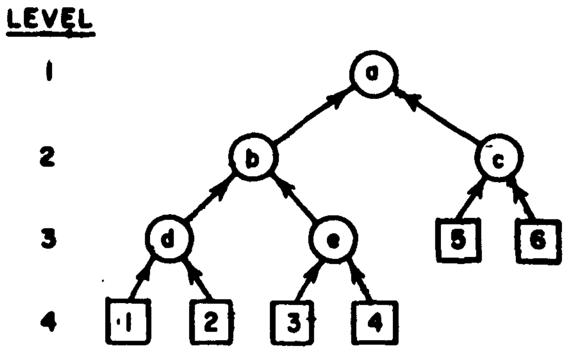
2. Greatest lower bound

It is noted in Figure 8A that all levels of the tree, exclusive of the first level, are deficient levels. Thus level 2 has only one of two possible proper nodes, level 3 has one of four possible proper nodes, etc. The sum $\sum L_i$ can be minimized by rearranging the paths in such a way as to eliminate one or more of the deficient levels. By placing two proper nodes on level 2, as shown in Figure 8B, the total length of all paths is reduced to 16: $\sum L_6'' = 3 + 3 + 3 + 3 + 2 + 2 = 16$. The number of levels is also reduced from six to four. L'' is minimum and hence L'' is the greatest lower bound of the sum under consideration.

It is also apparent from Figure 8B that a rearrangement of the branches at the nodes will give an equivalent $\sum L_i$ thus giving rise to a class of trees whose total path lengths are minimum. Interchanging nodes b and c in Figure 8B, for example, will give a total of path lengths equal to 16 which was previously found to be the minimum.



A. MAXIMUM $\sum L_i$ $(i = 6)$



B. MINIMUM $\sum L_i$ $(i = 6)$

FIG. 8 LEAST UPPER AND GREATEST LOWER BOUNDS FOR $\sum L_i$

In general, the number $\sum L_i$ can be minimized by doubling the number of proper nodes at each successive level, starting at the root, until the total amount of proper nodes is exhausted. For i blank nodes in a binary tree the minimum number of levels u is given by $2^{u-1} < 2i \leq 2^u$.

The minimum $\sum L_i$ can be obtained by enumeration as shown below.

Number Blank Nodes	$\sum L_i$
2	$2 = 1 \times 2$
3	$5 = 1 \times 2 + 1 \times 3$
4	$8 = 1 \times 2 + 2 \times 3$
5	$12 = 1 \times 2 + 2 \times 3 + 1 \times 4$
$\vdots$	$\vdots$
8	$24 = 1 \times 2 + 2 \times 3 + 4 \times 4$
9	$29 = 1 \times 2 + 2 \times 3 + 4 \times 4 + 1 \times 5$
$\vdots$	$\vdots$
16	$64 = 1 \times 2 + 2 \times 3 + 4 \times 4 + 8 \times 5$

etc.

The general formula is

$$\min \sum L_1 = 2 (i - 1) + n (i - 2^n) + \sum_{j=1}^{n-1} j2^j, \quad 2^n < i \leq 2^{n+1}.$$

The above discussion has not considered the contents of the blank nodes. Let us now assign a weight to each blank node, the weight being the frequency of symbol w_i of symbol set W. This weight will be designated f_i. Without loss of generality, the weights can be normalized to give the probability distribution of W.

We now wish to generate a binary tree which will render minimum $\sum p_i L_i$, minimum $\sum L_i$, and minimum L_{max}. Huffman's algorithm minimizes $\sum p_i L_i$ by ordering the frequencies (or probabilities) in ascending order, combining the two blank nodes of lowest weight, merging the combined frequency in the remaining set of ordered frequencies, and repeating the process for N - 1 combinations.

Because of the constraint on combining the two lowest weights the tree will have the property that the minimum weight of a node on the j-th level can never be less than the maximum weight of a node on the (j + 1)-st level.

The procedure previously described minimizes the sum of the lengths of all paths in a tree by minimizing the deficiency of each level. A pure proper node on level k contributes a zero deficiency to level k + 1; a mixed proper node contributes a deficiency of one; and a regular proper node contributes a deficiency of two. Hence a merger rule is desired which will insure a maximum number of proper nodes at the upper levels of the tree (those nearest the root) by providing, when a choice is available, formation first of a pure proper node, then a mixed proper node, and last a regular proper node. This will have the effect of assigning deficiencies arising from the frequency distribution to the lower levels of the tree.

Let us assign weights f_1 and f_2 to the two lowest frequencies (blank nodes) and designate their sum as $f_1^* = f_1 + f_2$. The combined frequency f_1^* will form a regular proper node and will appear on the (k - 1)-st level of the tree, where k is the bottom level furthest from the root. The combined frequency is compared with f_4, the second blank node of the next pair of blank nodes. One of three possibilities exist:

(i) $f_1{*} < f_4$, (ii) $f_1{*} = f_4$, (iii) $f_1{*} > f_4$.

(i) If $f_1{*} < f_4$, there is no choice and the combination rule requires that $f_1{*}$ be combined with f_3 to form the mixed proper node $f_{1,3}^{*}$ lying on level k - 2. $f_{1,3}^{*}$ will contribute a deficiency of one to level k - 1.

(ii) If $f_1{*} = f_4$, two alternatives are available for the next combination. Combining $f_1{*} + f_3$ will again form a mixed proper node $f_{1,3}^{*}$ lying on level k - 2 and contributing a deficiency of one to level k - 1. Combining $f_3 + f_4 = f_2{*}$ will form a regular proper node which, depending upon the frequency distribution, can lie on level k - 1 together with node $f_1{*}$. The combination $f_1{*} + f_3$ requires placing $f_{1,3}^{*}$ on level k - 2 thus guaranteeing a deficiency of one to level k - 1, irrespective of the frequency distribution. The combination $f_3 + f_4 = f_2{*}$ contributes a deficiency of two but allows the possibility of assigning the deficiency to the k-th level rather than the (k - 1)-st level.

If $f_1{*} = f_6$, again two alternatives are available for the next combination. Combining $f_1{*}$ with f_5 will create a mixed proper node which can assign a deficiency of one to the next lower level whereas combining $f_5 + f_6 = f_3{*}$ allows node $f_1{*}$ to lie on the same level and contribute its deficiency of two to the lowest level.

In general, merging a combined frequency below a set of frequencies with equal weight will assign deficiencies in levels of a tree to the lowest levels and will therefore minimize the sum of the path lengths. This merging rule will be called bottom merging. Placing a combined frequency above a set of frequencies of equal weight will be called top merging.

(iii) If $f_1{*} > f_4$, $f_1{*}$ is merged in the frequency list in its proper order. If $f_1{*}$ is distinct, it will be merged either above or below frequence f_i

depending upon whether $f_1{}^*$ is less than or greater than f_i. However, if $f_1{}^* = f_i$, it may be merged either above or below f_i without violating the ordering of the list. The two methods of merging establish two orderings each of which can incorporate $f_1{}^*$ in two combinations depending upon the way the frequencies are paired.

TOP MERGING	BOTTOM MERGING
$\begin{array}{l} \left.\begin{array}{l} f_{i-1} \\ f_1{}^* \end{array}\right\} T_1 \\ f_i \\ f_{i+1} \end{array}$	$\begin{array}{l} f_{i-1} \\ \left.\begin{array}{l} f_i \\ f_1{}^* \end{array}\right\} B_1 \\ f_{i+1} \end{array}$
$\begin{array}{l} f_{i-1} \\ \left.\begin{array}{l} f_1{}^* \\ f_i \end{array}\right\} T_2 \\ f_{i+1} \end{array}$	$\begin{array}{l} f_{i-1} \\ f_i \\ \left.\begin{array}{l} f_1{}^* \\ f_{i+1} \end{array}\right\} B_2 \end{array}$

It can be seen that in merge T_1, $f_1{}^*$ is combined with f_{i-1} and the combination requires that the node f_{i-1} lie on the same level, k - 1, as does $f_1{}^*$. Since no node on this level can have a weight greater than any node on the (k - 2)-nd level, the node formed from the combination $f_i + f_{i+1}$ will lie on the (k - 2)-nd level or on a higher level (k - j), $j > 2$. In the latter case it is possible for a deficiency to be assigned to the (k - j - 1)-st level depending on the frequency distribution. This deficiency will be propagated through the lower levels of the tree.

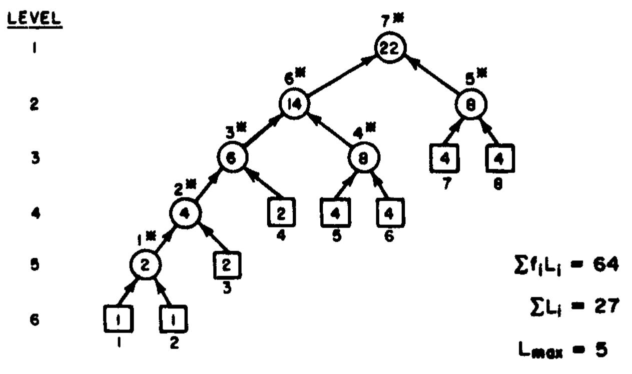
In merges T_2 and B_1, $f_1{}^*$ is combined with f_i which must lie on the same level as $f_1{}^*$. The node formed with f_{i+1} as one constituent will lie on the same level as $f_1{}^* + f_i$, namely, level k - 2, or on a higher level depending upon the frequency distribution. Again it is possible for a deficiency to be assigned to the (k - j - 1)-st level in the latter case.

In the B_2 merge, f_1* is combined with f_{i+1} which must lie on the same level as f_1*. The node formed from $f_{i-1} + f_i$ will either lie on the same level as the node formed from $f_1* + f_{i+1}$ or one level below. However, since the level below is the k-th or bottom level, only the blank nodes f_i and f_{i-1} can lie on this level. Hence no deficiency can be assigned to the (k - 1)-st or any higher level.

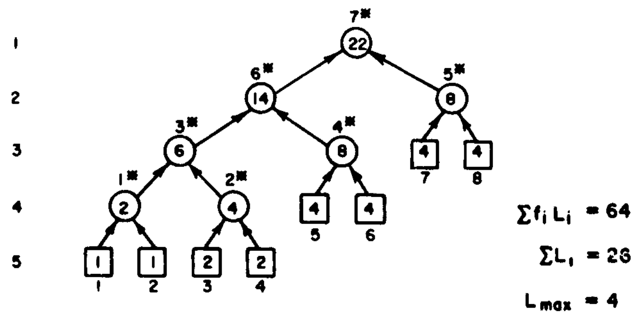
The same reasoning holds true for merging any combination of two frequencies in an ordered list. Whenever a choice is available, bottom merging will insure that the minimum sum of all path lengths is achieved by assigning deficiencies to the lower levels of the tree and filling the upper levels with pure proper nodes.

The variation in $\sum L_i$ using three merging rules for eight blank nodes is illustrated in Figure 9. It is noted that $\sum p_i L_i$ is constant for all three rules and that top merging results in the largest value of L_{max}, the longest path. The bottom merging rule results in minimum $\sum L_i$ and minimum L_{max}.

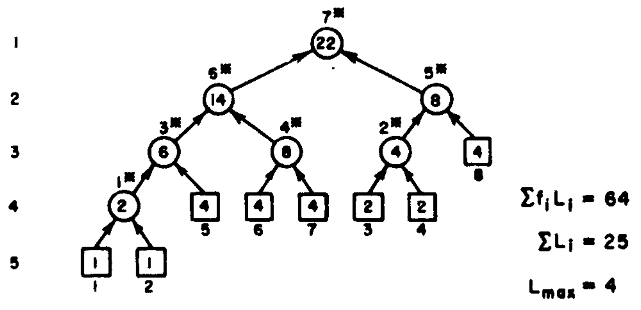
A computer program for generating an optimum encoding based upon Huffman's combinatorial method and which minimizes both L_{max} and $\sum L_i$ is now described. Input to the system consists of a dictionary of words to be encoded and a list of their associated frequencies. The procedure for constructing the codes consists in computing, from the list of frequencies, the number of words assigned codes of each of a number of code lengths in an optimal distribution. These values are used to transform the rank i of symbol w_i in the dictionary into the associated prefix code C_i.



A. TOP MERGING



B. RANDOM MERGING



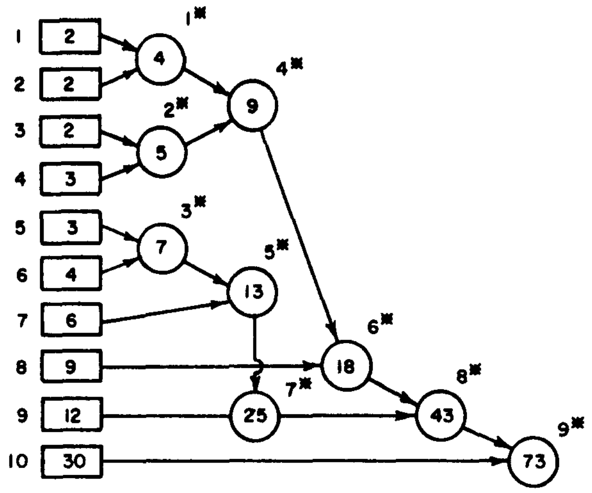
C. BOTTOM MERGING

FIG. 9 SUMMATION OF PATH LENGTHS WITH DIFFERENT MERGING RULES

The encoding is generated in four steps: 1) preparation of rank-frequency table, 2) generation of frequency tree, 3) determination of code breakpoints, and 4) assignment of codes. In the first step, the dictionary entries are sorted on frequency and are ordered in ascending frequency. A rank number (ties ignored) is assigned to each frequency for identification. Following the rule of adding the lowest two frequencies and bottom merging the combined frequency in the diminished frequency table, a frequency tree is generated. (Detailed flow charts and program descriptions are included in Section IX.)

The procedure is illustrated in Figure 10. The frequency tree for the given frequencies is shown in 10 A. The equivalent computer routine is shown in 10B. The program starts by adding the two lowest frequencies and placing the combined frequency and a starred rank number designating the combination in temporary storage as shown in pass 1 of Figure 10 B. The starred number designates a node of the tree. The frequencies, R_1 and R_2, that enter into a combination M^* (node) are identified by their rank numbers and are stored in a node table. In each pass, successive pairs of frequencies can be combined provided that the greater frequency of a pair is less than the sum of the initial combination in that pass. This check is made to insure that the rule of adding the two lowest frequencies is followed while permitting more than one combination in a pass. The combined frequencies obtained from the combined table are merged in the diminished rank-frequency table, placing starred frequencies after equal but unstarred frequencies. The process is then repeated.

Given N frequencies, the process will terminate when N-1 nodes have been formed. The rank-frequency table will be eliminated and the node



A. FREQUENCY TREE

L_i	
5	11111
5	11110
5	11101
5	11100
5	11011
5	11010
4	1100
3	101
3	100
1	0

$\sum L_i = 41$

C. CODE ASSIGNMENT

PASS: 1

RANK-FREQUENCY TABLE

R	f
1	2
2	2
3	2
4	3
5	3
6	4
7	6
8	9
9	12
10	30

COMBINED FREQUENCY

1*	4
2*	5

NODE TABLE

M	R_1	R_2
1*	1	2
2*	3	4

PASS: 2

R	f
5	3
6	4
1*	4
2*	5
7	6
8	9
9	12
10	30

3*	7
4*	9

M	R_1	R_2
1*	1	2
2*	3	4
3*	5	6
4*	1*	2*

PASS: 3

R	f
7	6
3*	7
8	9
4*	9
9	12
10	30

5*	13
6*	18

M	R_1	R_2
1*	1	2
2*	3	4
3*	5	6
4*	1*	2*
5*	7	3*
6*	8	4*

PASS: 4

R	f
9	12
5*	13
6*	18
10	30

7*	25

M	R_1	R_2
1*	1	2
2*	3	4
3*	5	6
4*	1*	2*
5*	7	3*
6*	8	4*
7*	9	5*

PASS: 5

R	f
6*	18
7*	25
10	30

8*	43

M	R_1	R_2
1*	1	2
2*	3	4
3*	5	6
4*	1*	2*
5*	7	3*
6*	8	4*
7*	9	5*
8*	6*	7*

PASS: 6

R	f
10	30
8*	43

9*	73

M	R_1	R_2
1*	1	2
2*	3	4
3*	5	6
4*	1*	2*
5*	7	3*
6*	8	4*
7*	9	5*
8*	6*	7*
9*	10	8*

B. COMBINE AND MERGE PROCEDURE

FIG. 10 COMPUTER GENERATION OF OPTIMUM EXHAUSTIVE PREFIX ENCODING. WITH MINIMUM $L_{max.}$ AND MINIMUM $\sum L_i$

table will be generated with N-1 entries. The six passes required for the example are shown in Figure 10 B.

The number of binary digits assigned to a code is the number of nodes in the branch originating at the initial frequency (blank node) and terminating at the (N-1)th combination (root). A count can be kept of the nodes in each branch during the generation of the node table or an iterative procedure can trace the nodes for each word frequency from the node table. However, a much simpler routine can be used to determine the points in the node table where the set of codes of length L change to codes of length L + 1. In one pass of the node table, all the breakpoints can be obtained.

The breakpoint routine is shown in the flow chart of Chart 3. M_i^* is a node formed from the combination of R_1 and R_2 ($\text{frequency}_{R_1} \leq \text{frequency}_{R_2}$). An unstarred R designates a blank node; a starred R designates a proper node. M_1 designates the node to which a search is directed by an R^* located at M_2, the location of the last breakpoint. The routine, in effect, divides the node table into a number of segments and then searches each segment for an unstarred R within this segment.

Given the optimum structure function of the encoding obtained from the breakpoint routine, a canonical encoding is generated in which the numerical values of the codes are monotone increasing and each code has the smallest possible numerical value consistent with the requirement that the code be an improper prefix.

Let the set of codes of length L be called a B - set. The breakpoint routine has determined the number of B - sets in the encoding and the number of codes in each B - set. A computer routine generates codes by assigning 0 to the first B - set (L = 1). Subsequent codes are assigned according to the rules:

1. Add binary one to the preceding code until the next B-set is reached;
2. Add binary 10 for the first code of a new B-set and return to step 1;
3. If a B-set is null, add 0 to the preceding code and return to step 1.

The set of codes thus generated is dense and storage efficiency is 100 percent since it is possible to use every successive storage location within a region in the enciphering and deciphering processes.

The canonical encoding for the example is shown in Figure 10 C. It is noted that the set of codes of length $L = 2$ is a null set. In most encodings, the set of codes of length $L = 1$ will be a null set, although a null set can occur for any value of L depending upon the frequency distribution of the words. A succession of null sets may be encountered in an exhaustive prefix encoding.

The summation of binary digits in all codes of the encoding using the top merging rule is 42 compared to 41 in the example obtained by employing the bottom merging rule; the maximum length code is 6 by top merging as compared to 5 in the example.

One of the advantages of using the above computer routine is that the only information required is the frequency distribution of the symbols to be encoded. It is neither necessary to compute the minimum cost, i.e., $-\sum p_i \log_2 p_i$, as a starting point in the integer programming method nor to determine the value of the maximum length code.

C. Encoding of Experimental Dictionary

The entropy of the experimental dictionary is 8.884 binary digits per dictionary entry. The average length of the 5,208 codes in the dictionary encoding is 8.920 binary digits per entry. Defining the efficiency E of an encoding as

$$E = \frac{-\sum p_i \log_2 p_i}{\sum p_i L_i} ,$$

the efficiency of the dictionary encoding is $\frac{8.884}{8.920} = 0.996$.

Whereas the average length of a word token in the experimental dictionary is 3.950 characters, the average length of all 5,208 entries including numerals, punctuation symbols, and suffixes is 3.541 characters per symbol. The information content of the encoded dictionary is, therefore, $\frac{8.920}{4.541} = 1.964$ binary digits per character, where one character was added to the average length of entry for a space.

Code lengths of the encoding ranged from 4 to 17 binary digits in the following distribution:

Code Length (Binary Digits)	Number Codes	Code Length (Binary Digits)	Number Codes
1	-	10	91
2	-	11	134
3	-	12	278
4	3	13	519
5	1	14	908
6	6	15	1,241
7	14	16	1,647
8	24	17	302
9	40	Total	5,208

A total of 75,983 binary digits were required for the complete encoding.

The encoding for the letter mode averaged 4.09 binary digits per letter as compared with a computed entropy of 4.03 binary digits per letter.

D. Number of EPE's For Given L_{max}.

An EPE satisfies the relation

$$2^{k-1} B_1 + 2^{k-2} B_2 + \ldots + B_k = 2^k = \sum_{i=1}^{k} 2^{k-i} B_i \qquad (1)$$

where $k = L_{max}$, the length of the longest code, B_i is the number of codes selected from the sequences of length i (i = 1, 2, . . . , k), and $\sum_{i=1}^{k} B_i = N$, the total number of codes in the encoding. The B_i are called the encoding structure functions.

Recalling that a B-set is the set of all sequences of channel letters that can be formed of length i, there are 2^i sequences in a B-set for the binary channel. When i = k, the number of sequences is 2^k which is the maximum number of codes obtainable for an EPE with a given L_{max}.

Each sequence that is selected as a code from the set of codes of length k - j ($1 \leq j < k$) is a proper prefix of 2 sequences of the set of codes of length (k - j + 1), 2^2 sequences of the set of length (k - j + 2), to 2^j sequences of the set of length k. Since by definition no code in an EPE can be the prefix of any other code, a selected code must be an improper prefix, and hence the sequences of which it would be a prefix must be eliminated from the encoding. The number of sequences eliminated in the encoding is, therefore, $2^{k-j} B_i$. This holds true for all B-sets but the set of length k in which every selected sequence is an improper prefix. Since there are 2^k possible sequences in the encoding, the summation of $2^{k-i} B_i$ over all values of i equals 2^k.

A canonical encoding for an EPE with $L_{max} = 4$, $N = 9$, and a structure function of 0 - 1 - 4 - 4 is illustrated in Figure 11.

The EPE's for any value of L_{max} can be generated by using equation (1). The six EPE's for $L_{max} = 3$ together with their respective codes are shown in Table 9 . For $L_{max} = 4$, 26 EPE's can be generated as shown in Table 10.

The maximum number of codes obtainable from an EPE of maximum length k is 2^k as noted previously. This encoding is of fixed length and is the binary sequence of numbers from zero through 2^k. The maximum value of B_{k-1} is $2^{k-1} - 1$, for if all sequences of this set were used, the EPE would be transformed into one of maximum length k - 1. Other EPE's are generated by varying the value of B_{k-1} from 1 through its maximum value while the value of B_k is successively decremented by 2 in accordance with equation (1). The value of B_{k-2} is similarly varied between 1 and its maximum $2^{k-2} - 1$ while all permutations on B_{k-1} and B_k are formed. This process is repeated until an encoding of k + 1 codes is reached. It is easy to see that this is the minimum number of codes obtainable from an EPE of length k.

A method for calculating the total number of EPE's that can be generated for any value of L_{max} is now described. Let EPE_k designate this number.

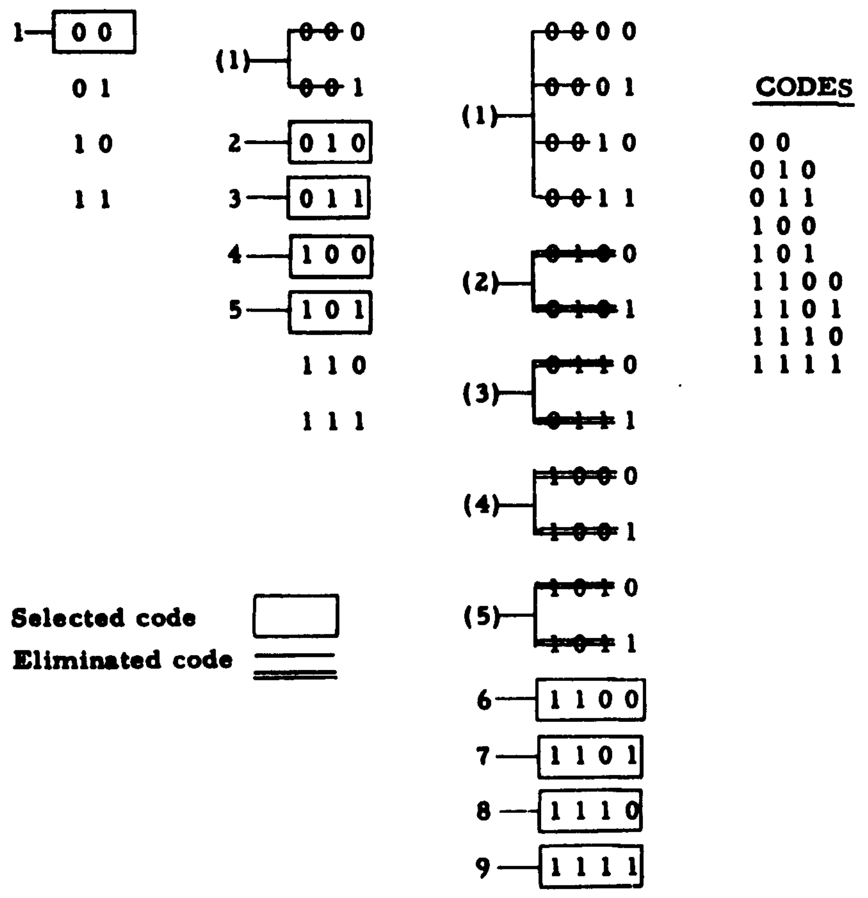


FIGURE 11 EXHAUSTIVE PREFIX ENCODING, $L_{max} = 4$, $N = 9$

TABLE 9. EXHAUSTIVE PREFIX ENCODINGS, $L_{MAX} = 3$

ENCODING	NO. CODES OF LENGTH			TOTAL CODES
	1 BIT	2 BITS	3 BITS	
A	0	0	8	8
B	0	1	6	7
C	0	2	4	6
D	0	3	2	5
E	1	0	4	5
F	1	1	2	4

ENCODING A	ENCODING B	ENCODING C
0 0 0	0 0	0 0
0 0 1	0 1 0	0 1
0 1 0	0 1 1	1 0 0
0 1 1	1 0 0	1 0 1
1 0 0	1 0 1	1 1 0
1 0 1	1 1 0	1 1 1
1 1 0	1 1 1	
1 1 1		

ENCODING D	ENCODING E	ENCODING F
0 0	0	0
0 1	1 0 0	1 0
1 0	1 0 1	1 1 0
1 1 0	1 1 0	1 1 1
1 1 1	1 1 1	

TABLE 10. EXHAUSTIVE PREFIX ENCODINGS, $L_{MAX} = 4$

ENCODING	NO. CODES OF LENGTH				TOTAL CODES
	1 BIT	2 BITS	3 BITS	4 BITS	
1	0	0	0	16	16
2	0	0	1	14	15
3	0	0	2	12	14
4	0	0	3	10	13
5	0	0	4	8	12
6	0	0	5	6	11
7	0	0	6	4	10
8	0	0	7	2	9
9	0	1	0	12	13
10	0	1	1	10	12
11	0	1	2	8	11
12	0	1	3	6	10
13	0	1	4	4	9
14	0	1	5	2	8
15	0	2	0	8	10
16	0	2	1	6	9
17	0	2	2	4	8
18	0	2	3	2	7
19	0	3	0	4	7
20	0	3	1	2	6
21	1	0	0	8	9
22	1	0	1	6	8
23	1	0	2	4	7
24	1	0	3	2	6
25	1	1	0	4	6
26	1	1	1	2	5

For k = 3,

$$EPE_3 = \sum_{i=1}^{2^{k-2}} 2i = 6.$$

For k = 4,

$$EPE_4 = \sum_{i=1}^{2^{k-2}} 2i + \sum_{i=1}^{2^{k-2}-2} 2i = 26.$$

For k = 5,

$$EPE_5 = \left[\sum_{i=1}^{2^{k-2}} 2i + \sum_{i=1}^{2^{k-2}-2} 2i + \sum_{i=1}^{2^{k-2}-2\cdot 2} 2i + \sum_{i=1}^{2^{k-2}-2\cdot 3} 2i\right]$$

$$+ \left[\sum_{i=1}^{2^{k-2}-2\cdot 2} 2i + \sum_{i=1}^{2^{k-2}-2\cdot 3} 2i\right] = 166.$$

Using the symbol $\overset{n}{\sum\textcircled{2}}$ to represent the summation of all values of even numbers from 2 through n, EPE_5, for example, can be written as

$$EPE_5 = \left[\overset{16}{\sum\textcircled{2}} + \overset{12}{\sum\textcircled{2}} + \overset{8}{\sum\textcircled{2}} + \overset{4}{\sum\textcircled{2}}\right] + \left[\overset{8}{\sum\textcircled{2}} + \overset{4}{\sum\textcircled{2}}\right] = 166.$$

A table showing the values of n generated from the permutations on k = 3 to k = 7 is shown in Table 11.

The summation of the $\overset{n}{\sum\textcircled{2}}$ values shown in Table 11 is expedited by collecting like terms, as shown below for k = 5:

$$\begin{aligned}
EPE_5 = 16 + 14 + {} & 12 + 10 + 8 + 6 + 4 + 2 + \\
& 12 + 10 + 8 + 6 + 4 + 2 + \\
& \phantom{12 + 10 + {}} 8 + 6 + 4 + 2 + \\
& \phantom{12 + 10 + 8 + 6 + {}} 4 + 2 + \\
& \phantom{12 + 10 + {}} 8 + 6 + 4 + 2 + \\
& \phantom{12 + 10 + 8 + 6 + {}} 4 + 2 \\
\hline
= 1\cdot 16 + 1\cdot 14 + 2\cdot 12 + 2\cdot 10 + 4\cdot 8 + 4\cdot 6 + 6\cdot 4 + 6\cdot 2 = 166
\end{aligned}$$

TABLE 11

ENCODING PERMUTATIONS $L_{MAX} = 3, 4, \ldots, 7$

L_{max}	VALUE OF n IN SUMMATION $\sum_{\textcircled{2}}^{n}$																									
3	4																									
4	8																									
	4																									
5	16																									
	12																									
	8	8																								
	4	4																								
6	32																									
	28																									
	24	24																								
	20	20																								
	16	16	16		16																					
	12	12	12		12																					
	8	8	8	8	8	8																				
	4	4	4	4	4	4																				
7	64																									
	60																									
	56	56																								
	52	52																								
	48	48	48						48																	
	44	44	44						44																	
	40	40	40	40					40	40																
	36	36	36	36					36	36																
	32	32	32	32	32				32	32	32				32						32					
	28	28	28	28	28				28	28	28				28						28					
	24	24	24	24	24	24			24	24	24	24			24	24					24	24				
	20	20	20	20	20	20			20	20	20	20			20	20					20	20				
	16	16	16	16	16	16	16		16	16	16	16	16		16	16	16		16		16	16	16		16	
	12	12	12	12	12	12	12		12	12	12	12	12		12	12	12		12		12	12	12		12	
	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4

The coefficients, each of which is used twice, of the terms in the summation EPE_k, form a series:

0 1 2 4 6 10 14 20 26 36 46 60 74 94 . . .

The values of the coefficients are:

$$a_{i_{ODD}} = a_{i-1} + a_{\frac{i+1}{2}}$$

$$a_{i_{EVEN}} = a_{i-1} + a_{\frac{i}{2}}$$

Given the first two coefficients of the series, $a_0 = 0$ and $a_1 = 1$, the remaining coefficients can be generated recursively. The first 64 coefficients are listed in Table 12.

In a general expression,

$$EPE_k = a_1 (2^{k-1}) + a_1 (2^{k-1}-2) + a_2 (2^{k-1}-2\cdot 2) + a_2 (2^{k-1}-2\cdot 3)$$

$$+ \dots + a_{2^{k-3}} (2^2) + a_{2^{k-3}} (2). \qquad (2)$$

Since the maximum number of encodings associated with the first term of the expression 2^{k-1} is known, and there are 2^{k-3} coefficients in the series (each coefficient is used twice), the number of EPE's for any value of L_{max} can be readily calculated from Table 12 and its extension.

Total EPE's for values of k from 3 through 9 are given below.

k	Total EPE
3	6
4	26
5	166
6	1,626
7	25,510
8	664,666
9	29,559,718

TABLE 12

COEFFICIENTS OF SERIES

TERM	COEFF.	TERM	COEFF.	TERM	COEFF.	TERM	COEFF.
1	1	17	202	33	1828	49	8350
2	2	18	238	34	2030	50	9042
3	4	19	284	35	2268	51	9828
4	6	20	330	36	2506	52	10614
5	10	21	390	37	2790	53	11514
6	14	22	450	38	3074	54	12414
7	20	23	524	39	3404	55	13428
8	26	24	598	40	3734	56	14442
9	36	25	692	41	4124	57	15596
10	46	26	786	42	4514	58	16750
11	60	27	900	43	4964	59	18044
12	74	28	1014	44	5414	60	19338
13	94	29	1154	45	5938	61	20798
14	114	30	1294	46	6462	62	22258
15	140	31	1460	47	7060	63	23884
16	166	32	1626	48	7658	64	25510

It is interesting to note that the summation of encodings for values of k is one of the coefficients in the series. Thus the fourth coefficient, 6, is EPE_3, the eighth coefficient, 26, is EPE_4, the 16th coefficient, 166, is EPE_5, etc. In general, the (2^{k-1})-th coefficient is the EPE_k.

The (2^{k-1})-th coefficient is an even term of the series and hence

$$EPE_k = a_{2^{k-1}} = a_{2^{k-1}-1} + a_{\frac{2^{k-1}}{2}}$$

$$= a_{2^{k-1}-1} + a_{2^{k-2}}.$$

The coefficient $a_{2^{k-2}}$ is the summation of EPE_{k-1} and the coefficient $a_{2^{k-1}-1}$ is the number of additional encodings that are generated by the permutations on the B_i's made possible by the addition of one digit to the previous maximum code length. For example,

$$EPE_5 = a_{16} = a_{15} + a_8 = 140 + 26 = 166.$$

The 26 codes are the maximum possible with $L_{max} = 4$ while 140 additional codes are made available by extending L_{max} to 5 digits.

E. Distribution of Number of Codes in Encodings

The previous discussion has been concerned with the total number of encodings that can be generated for any value of L_{max}. It is also of interest to determine the distribution of the number of codes in these encodings, i.e., given that EPE_k = M encodings, what number of encodings contain k + 1, k + 2, ..., 2^k codes respectively.

Let M_j (j = 1, 2, ..., EPE_k) be the number of codes in the j-th encoding obtainable for a given value of L_{max}. It will be recalled that $M_j = \sum_{i=1}^{k} B_{i_j}$. The B_{i_j} for the 26 encodings of k = 4 are listed in Table 10. In the eight permutations on B_k and B_{k-1}, the number of codes decreases by one in each successive encoding. When B_{k-2} is made equal to 1 (encoding 9 of Table 10), the 2^k codes of encoding 1 are initially decreased by 3 and are further decremented by 1 for each of the six encodings obtained as B_{k-1} and B_k are varied. A similar process is followed for each set of four and two encodings. Changing B_{k-3} to 1 (encoding 21 of Table 10), M_1 is diminished by 7 and successive encodings are diminished as above.

The distribution of the number of codes in the encodings for any value of L_{max} can be worked out in this manner, totalling the M_j of equal value without listing the structure function of each encoding.

The distribution of codes in the 166 EPE's obtained for k = 5 is shown in Table 13. From the table it is seen, for example, that there are 12 encodings with 14 codes obtainable, 7 encodings that contain 20 codes, etc.

Given N, the number of codes required in an encoding, the range in the number of binary digits from which these codes can be generated is

$$k + 1 \leq N \leq 2^k$$

where k is the maximum number of digits in the encoding. The 27 symbols of the English alphabet (including space) and disregarding their relative frequencies can be encoded with from 5 to 26 binary digits.

F. Non-uniqueness of Optimal Encoding

The previous sections have demonstrated that given any value of N, the number of symbols to be encoded, there are a number of encodings containing N codes for each value of maximum length code between k + 1 and 2^k binary digits. It was also shown in describing top and bottom merging that a number of frequency trees can be generated that are equal in the value of $\sum p_i L_i$ although $\sum L_i$ and L_{max} can be varied.

TABLE 13

DISTRIBUTION OF CODES

EXHAUSTIVE PREFIX ENCODINGS FOR $L_{MAX} = 5$

		NO. CODES IN ENCODING																										
	Encodings	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6
$\overset{16}{\Sigma}\textcircled{2}$	16	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1											
	14				1	1	1	1	1	1	1	1	1	1	1	1	1	1										
	12							1	1	1	1	1	1	1	1	1	1	1	1									
	10										1	1	1	1	1	1	1	1	1	1								
	8													1	1	1	1	1	1	1	1							
	6																1	1	1	1	1	1						
	4																			1	1	1	1					
	2																						1	1				
$\overset{12}{\Sigma}\textcircled{2}$	12								1	1	1	1	1	1	1	1	1	1	1	1								
	10											1	1	1	1	1	1	1	1	1	1							
	8														1	1	1	1	1	1	1	1						
	6																	1	1	1	1	1	1					
	4																				1	1	1	1				
	2																							1	1			
$\overset{8}{\Sigma}\textcircled{2}$	8															1	1	1	1	1	1	1	1					
	6																		1	1	1	1	1	1				
	4																					1	1	1	1			
	2																								1	1		
$\overset{4}{\Sigma}\textcircled{2}$	4																						1	1	1	1		
	2																									1	1	
$\overset{8}{\Sigma}\textcircled{2}$	8																1	1	1	1	1	1	1	1				
	6																			1	1	1	1	1	1			
	4																						1	1	1	1		
	2																									1	1	
$\overset{4}{\Sigma}\textcircled{2}$	4																							1	1	1	1	
	2																										1	1
TOTAL	166	1	1	1	2	2	2	3	4	4	5	6	6	7	8	9	11	11	11	12	11	10	11	10	7	6	4	1

It is no surprise, therefore, to find that an optimal encoding (minimum $\sum p_i L_i$) can be found for given N within several encodings of a stated L_{max} and also within encodings for different values of L_{max}. If the optimal encoding is defined as minimum $\sum p_i L_i$ and also minimum $\sum L_i$ and minimum L_{max}, the number of optimal encodings is reduced, but the optimum may still not be unique.

Consider, for example, all of the possible encodings that can be generated for N = 8 with the (top-bottom merge) frequencies shown in Figure 9 . A total of 16 encodings can be generated with from 3 to 7 binary digits as maximum code length. These encodings are tabulated in Table 14 It is noted that there are three encodings with minimum $\sum p_i L_i$ (indicated by :). Encoding 2 is optimum in the sense of minimizing all three parameters and is, in this case, unique.

TABLE 14

DISTRIBUTION OF $\sum p_i L_i$ AND $\sum L_i$

FOR ALL ENCODINGS, N = 8

Encoding No.	L_{max}	ENCODING STRUCTURE							$\sum p_i L_i$	$\sum L_i$
		B_{k-6}	B_{k-5}	B_{k-4}	B_{k-3}	B_{k-2}	B_{k-1}	B_k		
1	3					0	0	8	?.9700	32
2	4				0	1	5	2	2.9092^x	25*
3					0	2	2	4	2.9092^x	26
4					1	0	1	6	3.2728	28
5	5			0	2	3	1	2	2.9092^x	27
6				0	3	0	3	2	3.0002	28
7				0	3	1	0	4	3.0002	29
8				1	0	2	3	2	3.1820	29
9				1	0	3	0	4	3.1820	30
10				1	1	0	2	4	3.3638	31
11	6		0	3	1	1	1	2	3.0002	30
12			1	0	3	1	1	2	3.1820	31
13			1	1	0	3	1	2	3.3638	32
14			1	1	1	1	0	4	3.4548	34
15			1	1	1	0	3	2	3.4548	33
16	7	1	1	1	1	1	1	2	3.4548	35

^{x} minimum $\sum p_i L_i$

* minimum $\sum L_i$ and L_{max}

V. MESSAGE TRANSMISSION

A. Compression Ratio

The adaptive information transmission system has been tested on four samples ranging from 97 to 19,710 word tokens. Input to the system was either paper or magnetic tape with six binary digits per input character. The compression ratio and information content was determined for each sample together with a word analysis.

The compression ratio is defined as

$$\text{compression} = 1 - \frac{\text{output binary digits}}{\text{input binary digits}}$$

and represents the savings achieved by the minimum redundancy encoding and enciphering. The compression ratio may also be considered as the redundancy that was removed from a message.

Since the compression ratio is dependent upon the number of binary digits per input character, the information content of a sample, expressed as

$$I = \frac{\text{total number input characters}}{\text{total output binary digits}}$$

is used as a measure which is independent of the input format.

Messages were first processed through the transmitter enciphering program which produced an output tape (paper or magnetic) bearing the compressed enciphered message The output tape was then reinserted into the computer and reprocessed through the receiver deciphering and decoding program in order to recover the original message.

A 1,635 word message excerpted from six articles appearing in the New York Times is shown in Flexowriter format in Figures 12A, 12B, and 12C. This input text was compressed and the recovered information as produced by the receiver program is shown in Figures 13A and 13B. No

THE NEW YORK TIMES
NEW YORK, MONDAY, DECEMBER 3, 1962

U.S. SEES RESPITE IN WORLD TENSION AS TALKS PROCEED -
CALM IS EXPECTED DURING LONG PARLEYS WITH SOVIET ON POST-CUBA ISSUES -
WEST EXAMINES STAND -
KENNEDY, MACMILLAN AND DE GAULLE TO REDEFINE BERLIN AND ARMS VIEWS -
BY MAX FRANKEL
SPECIAL TO THE NEW YORK TIMES -

WASHINGTON, DEC. 2 - THE ADMINISTRATION IS LOOKING AHEAD TO A LONG BUT PROBABLY INCONCLUSIVE SERIES OF EXPLORATORY TALKS WITH THE SOVIET UNION TO REDEFINE MAJOR DIPLOMATIC PROBLEMS IN THE WAKE OF THE CUBAN CRISIS.

IT EXPECTS AN UNCERTAIN CALM ON THE INTERNATIONAL SCENE DURING THESE DISCUSSIONS - A RESPITE FROM THE GREAT TENSIONS OF THE LAST TWO MONTHS, BUT SOME-THING FAR SHORT OF A RELAXATION OR ACCOMMODATION.

ALTHOUGH THE CUBAN SITUATION HAS NOT BEEN RESOLVED TO THE FULL SATISFACTION OF EITHER SIDE, THE FEELING HERE IS THAT THE REMOVAL OF SOVIET JET BOMBERS FROM CUBA, WHICH IS IMMINENT, WILL ALMOST EXHAUST THE POSSIBILITIES FOR NEGOTIATION IN THE IMMEDIATE SITUATION.

THE ADMINISTRATION EXPECTS THEN TO TURN TO THE PROBLEM OF THE EXISTENCE OF THE CASTRO REGIME. THIS MATTER WILL BE INFLUENCED CONSIDERABLY BY MOSCOWS RELATIONS WITH HAVANA.

THE RUSSIANS ARE BELIEVED TO BE EAGER TO REDUCE THEIR FINANCIAL COMMITMENTS TO CUBA, AND THIS LEADS OFFICIALS TO THE IDEA THAT, WHILE CUBA WILL REMAIN AN IMPORTANT PROBLEM, IT WILL BE ONE FOR THE UNITED STATES ALONE AND NOT FOR JOINT SOVIET-UNITED STATES ACTION OR AGREEMENT.

THE OFFICIALS THUS EXPECT EAST-WEST PARLEYS TO TURN ONCE MORE TO THE PROBLEMS OF BERLIN, A TREATY TO FORBID NUCLEAR TESTS AND A VARIETY OF OTHER DISARMAMENT MEASURES.

THE DOMINANT MOOD HERE SEEMS TO BE THAT IF THE RUSSIANS WANT SUCH RESPITE AND REAPPRAISAL OF INTENTIONS AND POSITIONS, THE UNITED STATES SHOULD COOPERATE.

BIG INDIAN FORCE ESCAPES CHINESE -
RADIO REPORTS 10,000 WERE AMONG 14,000 CUT OFF BY ENEMYS FLANKING DRIVE -
BY THOMAS F. BRADY
SPECIAL TO THE NEW YORK TIMES -

NEW DELHI, DEC. 2 - TEN THOUSAND INDIAN TROOPS TRAPPED BY A CHINESE COMMUNIST FLANKING DRIVE IN NORTHEAST INDIA WERE REPORTED TODAY TO HAVE MADE THEIR WAY BACK TO THE INDIAN LINES.

THE STATE-MANAGED ALL-INDIA RADIO SAID THE TROOPS WERE PART OF THE 14,000 SOLDIERS OF THE FOURTH DIVISION WHO WERE CUT OFF IN THE CHINESE DRIVE NORTH OF BOMDI LA, IN THE NORTH EAST FRONTIER AGENCY, NOV. 18.

THE INDIAN GOVERNMENT MAINTAINED SILENCE ON THE WITHDRAWAL OF THE CHINESE TROOPS FROM THE CONQUERED AREAS OF NORTHEAST INDIA, WHICH THE PEKING RADIO HAS ANNOUNCED IS UNDER WAY.

THE INDIAN DIVISION, TRAPPED IN THE PASSES NEAR INDIAS BORDER WITH BHUTAN, DESTROYED ALL ITS HEAVY EQUIPMENT INCLUDING TANKS TO PREVENT THEIR LOSS TO THE CHINESE, THE RADIO SAID. THE 10,000 MEN WHO HAVE RETURNED BROUGHT LIGHT ARMS WITH THEM, ACCORDING TO THE RADIO.

ADENAUER PARTY WEIGHING UNION WITH SOCIALISTS -
OPPOSITION IS APPROACHED BY HOUSING CHIEF IN BID TO END POLITICAL CRISIS -
BY SYDNEY GRUSON
SPECIAL TO THE NEW YORK TIMES -

Figure 12A INPUT TEXT OF NEW YORK TIMES ARTICLES

BONN, DEC. 2 - A COALITION BETWEEN WEST GERMANYS TWO MAJOR PARTIES HAS BEEN SUGGESTED AS A POSSIBLE SOLUTION OF THE GOVERNMENT CRISIS.

IT WAS DISCLOSED TODAY THAT PAUL LUCKE, HOUSING MINISTER IN CHANCELLOR ADENAUERS CARETAKER CABINET, HAD HELD SEVERAL TALKS WITH THE OPPOSITION SOCIAL DEMOCRATS ON BEHALF OF ONE FACTION IN DR. ADENAUERS PARTY, THE CHRISTIAN DEMOCRATIC UNION.

THE TALKS WERE HELD LAST WEEK AND THEY WERE SAID TO HAVE BEEN CONDUCTED WITH THE KNOWLEDGE OF BUT NOT AT THE INSTIGATION OF THE CHANCELLOR.

MR. LUCKES TALKS WITH THE SOCIALISTS WERE KEPT SECRET UNTIL TODAY. HE VISITED DR. ADENAUER FOUR OR FIVE TIMES LAST WEEK. IT WAS SAID THAT DURING THESE MEETINGS DR. ADENAUER WAS TRYING TO PERSUADE MR. LUCKE TO REPLACE FRANZ JOSEPH STRAUSS AS DEFENSE MINISTER. APPARENTLY, HOWEVER, THEY DISCUSSED MR. LUCKES MEETINGS WITH THE SOCIALISTS.

THE SOCIALISTS HAVE REACTED WITH GREAT CAUTION TO MR. LUCKES APPROACHES. THEY WERE REPORTED TO HAVE TOLD HIM THAT IF THE CHRISTIAN DEMOCRATS FORMALLY REQUESTED COALITION NEGOTIATIONS THEY WOULD MEET TO DISCUSS THE QUESTION.

BUT MR. LUCKE WAS ALSO TOLD THAT THE SOCIALISTS WANTED NO PART OF COALITION TALKS THAT WERE MEANT ONLY TO SOFTEN UP THE FREE DEMOCRATIC PARTYS TERMS FOR REJOINING THE GOVERNMENT.

THE DEPUTIES OF THE CHRISTIAN DEMOCRATIC UNION AND THEIR BAVARIAN ALLY, THE CHRISTIAN SOCIAL UNION, HAVE BEEN CALLED TO A MEETING TOMORROW TO DEBATE WHETHER THE SOCIALISTS SHOULD NOW BE APPROACHED OFFICIALLY. A NEGOTIATING SESSION THAT WAS SCHEDULED FOR TOMORROW BETWEEN DR. ADENAUER AND THE FREE DEMOCRATS HAS BEEN POSTPONED UNTIL TUESDAY.

SUDDEN OIL WEALTH CONFUSES LIFE IN LIBYA -
PARLIAMENT TO MEET TO DECIDE BEST WAY TO SPEND MONEY -
BY JAY WALZ
SPECIAL TO THE NEW YORK TIMES -

BENGHAZI, LIBYA, NOV. 28 - THE PARLIAMENT OF THIS CINDERELLA-POOR COUNTRY THAT SUDDENLY STUCK IT RICH WITH OIL WILL MEET DEC. 6 TO STUDY THE PROBLEMS ARISING FROM THE NEW WEALTH.

THE PARLIAMENT WILL CONVENE IN THE NEW $50,000,000 FEDERAL CITY OF BEIDA, 135 MILES NORTHEAST OF BENGHAZI. THE NEW CITY IS SPREAD AWKWARDLY OVER A HIGH, WINDSWEPT PLAIN THAT THROUGH THE CENTURIES HAS KNOWN WOODS OF SCRUB PINE AND THORN, BEDOUINS WITH THEIR FLOCKS, AND, FLEETINGLY DURING WORLD WAR II, THE FAST-MOVING MOTOR DIVISIONS OF FIELD MARSHALS MONTGOMERY AND ROMMEL.

BEIDA IS A SHOW OF NEW WEALTH, ALTHOUGH IT WAS STARTED WHILE THE INFANT COUNTRY WAS LIVING LARGELY ON FOREIGN AID. CONTRACTORS AND CARPENTERS HAVE BEEN UNABLE TO RUSH BEIDAS PARLIAMENT HOUSE TO COMPLETION IN TIME. BUT KING IDRIS I, LIBYAS 72-YEAR-OLD RULER, IS SO DETERMINED TO GET THE GOVERNMENT TO WORK IN BEIDA THAT THE PARLIAMENTS DEPUTIES WILL TEMPORARILY WORK AT THE DESKS OF THE LOCAL HIGH SCHOOL.

OIL MONEY IS NO LONGER A GLITTERING MIRAGE IN THE LIBYAN DESERT; IT IS POURING IN. THE FOUR-YEAR SEARCH FOR OIL BEGAN TO PAY OFF WITH THE COMMERCIALLY USEFUL WELLS IN 1959. NOW, INVESTMENTS BY 18 COMPANIES AND COMBINATIONS OF COMPANIES SHOW A REAL RESULT.

PIPELINES AND TERMINALS SEND 165,000 BARRELS A DAY TO THE WORLDS MARKETS FROM THE FIELDS, AROUND DAHRA, 100 MILES SOUTH OF TRIPOLI, AND AT ZELTEN, IN THE PROVINCE OF CYRENAICA, ON THE BORDER OF THE UNITED ARAB REPUBLIC.

JUST AS REAL AS THE OIL GUSHING THROUGH THE PIPELINES ARE THE GROWING PAINS LIBYA IS EXPERIENCING. THE PARLIAMENT, KING IDRIS HAS DETERMINED, MUST SHARE THE RESPONSIBILITY FOR USING THE WEALTH WISELY.

THORIUM IS FOUND IN NEW HAMPSHIRE -
DEPOSIT DOUBLES ESTIMATE OF ATOM FUEL IN U.S. -
BY JOHN A. OSMUNDSEN -

AN ASSESSMENT OF A LOW-GRADE THORIUM DEPOSIT IN THE MOUNTAINS OF NEW HAMPSHIRE HAS DOUBLED PREVIOUS ESTIMATES OF THE SIZE OF THIS NATIONS TOTAL RESERVE OF NUCLEAR FUEL.

SCIENTISTS AT RICE UNIVERSITY IN HOUSTON HAVE REPORTED FINDING AT LEAST TENS OF MILLIONS OF TONS OF THORIUM IN THE MOUNTAIN RANGE. THIS MULTIPLIED

Figure 12B INPUT TEXT OF NEW YORK TIMES ARTICLES

EARLIER ESTIMATES OF THE SIZE OF THE RESERVE OF THAT RADIOACTIVE ELEMENT BY A FACTOR OF NO LESS THAN 10.

THIS WOULD RAISE ESTIMATES OF THE THORIUM RESERVE IN THIS COUNTRY TO ABOUT EQUAL THE RESERVE OF URANIUM WHICH IS SAID TO BE ABOUT 90,000,000 TONS.

MINING THE THORIUM DEPOSIT WOULD NOT BE ECONOMICAL NOW. THE NUCLEAR ENGINEERING TECHNIQUES FOR USING THE METAL AS A FUEL IN AN ATOMIC REACTOR HAVE NOT YET BEEN WORKED OUT.

RESEARCH IS UNDER WAY, HOWEVER, TO FIND WAYS OF REDUCING THE COST OF EXTRACTING THE RADIO-ACTIVE ELEMENT FROM ITS ORE AND EVENTUALLY USING IT IN INDUSTRY.

DR. JOHN A. S. ADAMS, WHO DIRECTED THE RESEARCH TEAM THAT PRODUCED THE NEW ASSESSMENT OF THE THORIUM RESERVE, SAID IN A TELEPHONE INTERVIEW THAT THE ESTIMATE WAS 'PROBABLY VERY CONSERVATIVE.'

STOCKS EDGE OFF AS TRADING EASES -
SHORT COVERING AND PROFIT TAKING CONTRIBUTE TO DROP-AVERAGE FALLS BY 1.71 -
TURNOVER IS 4,571,460 -
LOSS ELICITS LITTLE SURPRISE ON WALL STREET - GAINS TOP DECLINES, 563-498 -
BY RICHARD RUTTER -

THE STOCK MARKET HAD A MILD DECLINE YESTERDAY AS TRADING TAPERED OFF.

THE MINOR SET-BACK WAS NEITHER SURPRISING NOR ALARMING TO MARKET FOLLOWERS AND OBSERVERS. IT FOLLOWED NOT ONLY THREE DAYS OF GAINS BUT A GENERAL ADVANCE THAT HAS PERSISTED FOR THE BETTER PART OF THE LAST FIVE WEEKS. IT WAS AN ADVANCE THAT SAW STOCKS, ON AVERAGE, POST THE BEST MONTHLY GAINS SINCE THE NEW YORK TIMES BEGAN TO COMPILE MARKET AVERAGES IN 1911. VOLUME FOR NOVEMBER WAS THE LARGEST FOR THE MONTH SINCE 1928.

YESTERDAY, BOTH PROFIT TAKING AND SHORT COVERING WAS EVIDENT.

STOCKS OPENED LOWER ON THE NEW YORK STOCK EXCHANGE, DRIFTED DOWN UNTIL ABOUT NOON AND THEN FIRMED. THE LATE RECOVERY WAS NOT QUITE SUBSTANTIAL ENOUGH TO ERASE ALL THE EARLIER LOSSES, BUT VARIOUS OIL, RAIL AND SPECIAL SITUATION ISSUES HAD GOOD GAINS.

THE LATEST BATCH OF ECONOMIC NEWS WAS ENCOURAGING, PERHAPS ENOUGH TO AVERT A LARGER LOSS. CONSTRUCTION CONTRACT AWARDS IN OCTOBER WERE REPORTED AT A RECORD, PRESAGING A CONTINUED BUILDING BOOM. MANUFACTURES NEW ORDERS ALSO REACHED A NEW HIGH, ALTHOUGH SALES DIPPED. AUTO PRODUCTION WAS AT THE HIGHEST LEVEL IN ALMOST THREE YEARS.

THE FACT THAT VOLUME CONTRACTED ON THE MARKET DIP WAS ALSO TAKEN AS A MORE OR LESS BULLISH DEVELOPMENT.

VOLUME ON THE NEW YORK STOCK EXCHANGE WAS 4,571,460 SHARES, DOWN FROM 5,810,000 ON THURSDAY. IT WAS THE FIRST TIME IN THE LAST SEVEN TRADING SESSIONS THAT TURNOVER HAD FALLEN BELOW THE 5,000,000-SHARE LEVEL.

UTILITIES TO LEAD IN OFFERINGS OF BONDS DURING COMING WEEK -

NEW ISSUES OF FIXED-INTEREST SENIOR SECURITIES EXPECTED NEXT WEEK WILL AGAIN BE LED BY UTILITY OFFERINGS. TOTAL MUNICIPAL AND CORPORATE DEBT SECURITIES SCHEDULED FOR SALE EXCEED $186,000,000, AGAINST $222,000,000 THIS WEEK. BOTH FIGURES ARE BASED ON ISSUES OF $1,000,000 OR MORE.

THE TOTAL OF CORPORATE AND MUNICIPAL DEBT ISSUES EXPECTED IN DECEMBER WAS PUT AT $647,000,000, COMPARED WITH ABOUT $693,000,000 IN THE 1961 PERIOD, ACCORDING TO THE COMMERCIAL AND FINANCIAL CHRONICLE, A TRADE PUBLICATION.

OF NEXT WEEKS SCHEDULED OFFERINGS, THE LARGEST COMPETITIVE SALE IS $60,000,000 OF CONSOLIDATED EDISON COMPANY OF NEW YORK BONDS, WHICH IS EXPECTED ON WEDNESDAY.

MAJOR OFFERINGS EXPECTED MONDAY INCLUDE $15,000,000 OF METROPOLITAN EDISON COMPANY BONDS; $9,000,000 OF FLORIDA DEVELOPMENT COMMISSION REVENUE BONDS $9,900,000 OF GARY, IND., SANITATION DISTRICT BONDS, AND $8,000,000 OF DALLAS COUNTY, TEX., BONDS.

Figure 12C INPUT TEXT OF NEW YORK TIMES ARTICLES

```
THE NEW YORK TIMFS NEW YORK, MONDAY, DECEMBER 3,1962 U.  S.  SEES RESPITE IN WORLD TFNSION AS TALKS PROCFED - CALM IS E
XPECTED DURING LONG PARLFYS WITH SOVIET ON POST - CUBA ISSUES - WEST FXAMINES STAND - KENNFDY, MACMILLAN AND DE GAULLE T
O REDEFINE BERLIN AND ARMS VIFWS - BY MAX FRANKEL SPECIAL TO THE NEW YORK TIMES - WASHINGTON, DEC.2 - THE ADMINISTRATION
 IS LOOKING AHEAD TO A LONG BUT PROBABLY INCONCLUSIVE SERIES OF EXPLORATORY TALKS WITH THF SOVIET UNION TO REDEFINE MAJO
R DIPLOMATIC PROBLFMS IN THE WAKE OF THE CUBAN CRISIS.  IT EXPECTS AN UNCFRTAIN CALM ON THF INTERNATIONAL SCENE DURING T
HESE DISCUSSIONS - A RESPITE FROM THE GREAT TENSIONS OF THE LAST TWO MONTHS, BUT SOME - THING FAR SHORT OF A RELAXATION
OR ACCOMMODATION.  ALTHOUGH THE CUBAN SITUATION HAS NOT BEEN RESOLVED TO THE FULL SATISFACTION OF EITHER SIDE, THF FEELI
NG HERE IS THAT THE REMOVAL OF SOVIET JET BOMBERS FROM CUBA, WHICH IS IMMINENT, WILL ALMOST EXHAUST THE POSSIBILITIES FO
R NEGOTIATION IN THE IMMEDIATE SITUATION.  THE ADMINISTRATION EXPECTS THEN TO TURN TO THE PROBLEM OF THE FXISTENCF OF TH
E CASTRO REGIME.  THIS MATTER WILL BE INFLUENCED CONSIDERABLY BY MOSCOWS RELATIONS WITH HAVANA.  THE RUSSIANS ARE BELIFV
ED TO BE EAGER TO REDUCE THEIR FINANCIAL COMMITMENTS TO CUBA, AND THIS LEADS OFFICIALS TO THE IDEA THAT, WHILE CUBA WILL
 REMAIN AN IMPORTANT PROBLFM, IT WILL BE ONE FOR THE UNITED STATES ALONE AND NOT FOR JOINT SOVIET - UNITED STATES ACTION
 OR AGREEMENT.  THF OFFICIALS THUS EXPECT EAST - WEST PARLEYS TO TURN ONCE MORE TO THE PROBLEMS OF BERLIN, A TREATY TO F
ORBID NUCLEAR TESTS AND A VARIETY OF OTHER DISARMAMENT MEASURES.  THE DOMINANT MOOD HERE SEEMS TO BE THAT IF THE RUSSIAN
S WANT SUCH RESPITE AND REAPPRAISAL OF INTENTIONS AND POSITIONS, THE UNITFD STATES SHOULD COOPERATE.  BIG INDIAN FORCE E
SCAPES CHINESE - RADIO REPORTS 10,000 WERE AMONG 14,000 CUT OFF BY ENEMYS FLANKING DRIVF - BY THOMAS F.  BRADY SPECIAL T
O THE NEW YORK TIMFS - NEW DELHI, DEC.2 - TEN THOUSAND INDIAN TROOPS TRAPPED BY A CHINESE COMMUNIST FLANKING DRIVF IN NO
RTHEAST INDIA WERF REPORTED TODAY TO HAVE MADE THEIR WAY BACK TO THE INDIAN LINES.  THF STATE - MANAGED ALL - INDIA RADI
O SAID THE TROOPS WERE PART OF THE 14,000 SOLDIERS OF THE FOURTH DIVISION WHO WERE CUT OFF IN THE CHINESE DRIVE NORTH OF
 BOMDI LA, IN THE NORTH FAST FRONTIER AGENCY, NOV.18.  THE INDIAN GOVERNMENT MAINTAINED SILENCF ON THE WITHDRAWAL OF THE
 CHINESE TROOPS FROM THE CONQUERED AREAS OF NORTHEAST INDIA, WHICH THE PEKING RADIO HAS ANNOUNCED IS UNDER WAY.  THE IND
IAN DIVISION, TRAPPED IN THE PASSES NEAR INDIAS BORDER WITH BHUTAN, DESTROYED ALL ITS HEAVY EQUIPMENT INCLUDING TANKS TO
 PREVENT THEIR LOSS TO THE CHINESE, THE RADIO SAID.  THE 10,000 MEN WHO HAVE RETURNED BROUGHT LIGHT ARMS WITH THEM, ACCO
RDING TO THE RADIO.  ADENAUER PARTY WEIGHING UNION WITH SOCIALISTS - OPPOSITION IS APPROACHED BY HOUSING CHIEF IN BID TO
 END POLITICAL CRISIS - BY SYDNEY GRUSON SPECIAL TO THE NEW YORK TIMES - BONN, DEC.2 - A COALITION BETWEEN WEST GERMANYS
 TWO MAJOR PARTIES HAS BEEN SUGGESTED AS A POSSIBLE SOLUTION OF THE GOVERNMENT CRISIS.  IT WAS DISCLOSED TODAY THAT PAUL
 LUCKE, HOUSING MINISTER IN CHANCELLOR ADENAUERS CARETAKER CABINET, HAD HELD SEVERAL TALKS WITH THE OPPOSITION SOCIAL DE
MOCRATS ON BEHALF OF ONE FACTION IN DR.  ADENAUERS PARTY, THE CHRISTIAN DEMOCRATIC UNION.  THE TALKS WERE HELD LAST WEFK
 AND THEY WERE SAID TO HAVE BEEN CONDUCTED WITH THE KNOWLEDGE OF BUT NOT AT THE INSTIGATION OF THE CHANCELLOR.  MR.  LUC
KES TALKS WITH THF SOCIALISTS WERE KEPT SECRET UNTIL TODAY.  HE VISITED DR.  ADENAUER FOUR OR FIVE TIMES LAST WEEK.  IT
WAS SAID THAT DURING THESE MEETINGS DR.  ADENAUER WAS TRYING TO PERSUADE MR.  LUCKE TO REPLACE FRANZ JOSEPH STRAUSS AS D
EFENSE MINISTER.  APPARENTLY, HOWEVER, THEY DISCUSSED MR.  LUCKES MEETINGS WITH THE SOCIALISTS.  THE SOCIALISTS HAVE REA
CTED WITH GREAT CAUTION TO MR.  LUCKES APPROACHES.  THEY WERE REPORTED TO HAVE TOLD HIM THAT IF THE CHRISTIAN DEMOCRATS
FORMALLY REQUESTED COALITION NEGOTIATIONS THEY WOULD MEET TO DISCUSS THE QUESTION.  BUT MR.  LUCKE WAS ALSO TOLD THAT TH
E SOCIALISTS WANTED NO PART OF COALITION TALKS THAT WERE MEANT ONLY TO SOFTEN UP THE FREE DEMOCRATIC PARTYS TERMS FOR RE
JOINING THE GOVERNMENT.  THE DEPUTIES OF THE CHRISTIAN DEMOCRATIC UNION AND THEIR BAVARIAN ALLY, THE CHRISTIAN SOCIAL UN
ION, HAVE BEEN CALLED TO A MEETING TOMORROW TO DEBATE WHETHER THE SOCIALISTS SHOULD NOW BE APPROACHED OFFICIALLY.  A NEG
OTIATING SESSION THAT WAS SCHEDULED FOR TOMORROW BETWEEN DR.  ADENAUER AND THE FREE DEMOCRATS HAS BEEN POSTPONED UNTIL T
UESDAY.  SUDDEN OIL WEALTH CONFUSES LIFE IN LIBYA - PARLIAMENT TO MEET TO DECIDE BEST WAY TO SPEND MONEY - BY JAY WALZ S
PECIAL TO THE NEW YORK TIMES - BENGHAZI, LIBYA, NOV.28 - THE PARLIAMENT OF THIS CINDERELLA - POOR COUNTRY THAT SUDDENLY
STUCK IT RICH WITH OIL WILL MEET DEC.6 TO STUDY THE PROBLEMS ARISING FROM THE NEW WEALTH.  THE PARLIAMENT WILL CONVENE I
N THE NEW $50,000,000 FEDERAL CITY OF BEIDA,135 MILES NORTHEAST OF BENGHAZI.  THE NEW CITY IS SPREAD AWKWARDLY OVER A HI
GH, WINDSWEPT PLAIN THAT THROUGH THE CENTURIES HAS KNOWN WOODS OF SCRUB PINE AND THORN, BEDOUINS WITH THEIR FLOCKS, AND,
 FLEETINGLY DURING WORLD WAR II, THE FAST - MOVING MOTOR DIVISIONS OF FIELD MARSHALS MONTGOMERY AND ROMMEL.  BEIDA IS A
SHOW OF NEW WEALTH, ALTHOUGH IT WAS STARTED WHILE THE INFANT COUNTRY WAS LIVING LARGELY ON FOREIGN AID.  CONTRACTORS AND
 CARPENTERS HAVE BEEN UNABLE TO RUSH BEIDAS PARLIAMENT HOUSE TO COMPLETION IN TIME.  BUT KING IDRIS I, LIBYAS 72- YEAR -
 OLD RULER, IS SO DETERMINED TO GET THE GOVERNMENT TO WORK IN BEIDA THAT THE PARLIAMENTS DEPUTIES WILL TEMPORARILY WORK
AT THE DESKS OF THE LOCAL HIGH SCHOOL.  OIL MONEY IS NO LONGER A GLITTERING MIRAGE IN THE LIBYAN DESERT! IT IS POURING I
N.  THE FOUR - YEAR SEARCH FOR OIL BEGAN TO PAY OFF WITH THE COMMERCIALLY USEFUL WELLS IN 1959.  NOW, INVESTMENTS BY 18
COMPANIES AND COMBINATIONS OF COMPANIES SHOW A REAL RESULT.  PIPELINES AND TERMINALS SEND 165,000 BARRELS A DAY TO THE W
ORLDS MARKETS FROM THE FIELDS, AROUND DAHRA,100 MILES SOUTH OF TRIPOLI, AND AT ZELTEN, IN THE PROVINCE OF CYRENAICA, ON
THE BORDER OF THE UNITED ARAB REPUBLIC.  JUST AS REAL AS THE OIL GUSHING THROUGH THE PIPELINES ARE THE GROWING PAINS LIB
YA IS EXPERIENCING.  THE PARLIAMENT, KING IDRIS HAS DETERMINED, MUST SHARE THE RESPONSIBILITY FOR USING THE WEALTH WISEL
Y.  THORIUM IS FOUND IN NEW HAMPSHIRE - DEPOSIT DOUBLES ESTIMATE OF ATOM FUEL IN U.  S.- BY JOHN A.  OSMUNDSEN - AN ASSE
SSMENT OF A LOW - GRADE THORIUM DEPOSIT IN THE MOUNTAINS OF NEW HAMPSHIRE HAS DOUBLED PREVIOUS ESTIMATES OF THE SIZE OF
THIS NATIONS TOTAL RESERVE OF NUCLEAR FUEL.  SCIENTISTS AT RICE UNIVERSITY IN HOUSTON HAVF REPORTED FINDING AT LEAST TEN
S OF MILLIONS OF TONS OF THORIUM IN THE MOUNTAIN RANGE.  THIS MULTIPLIED EARLIER ESTIMATES OF THE SIZE OF THE RESERVE OF
 THAT RADIOACTIVE ELEMENT BY A FACTOR OF NO LESS THAN 10.  THIS WOULD RAISE ESTIMATES OF THE THORIUM RESERVE IN THIS COU
NTRY TO ABOUT EQUAL THE RESERVE OF URANIUM WHICH IS SAID TO BE ABOUT 90,000,000 TONS.  MINING THE THORIUM DEPOSIT WOULD
NOT BE ECONOMICAL NOW.  THF NUCLEAR ENGINEERING TECHNIQUES FOR USING THE METAL AS A FUEL IN AN ATOMIC REACTOR HAVE NOT Y
```

Figure 13A OUTPUT TEXT OF NEW YORK TIMES ARTICLES

```
ET BEEN WORKED OUT.  RESEARCH IS UNDER WAY, HOWEVER, TO FIND WAYS OF REDUCING THE COST OF FXTRACTING THE RADIO - ACTIVE
ELEMENT FROM ITS ORE AND FVENTUALLY USING IT IN INDUSTRY.  DR.  JOHN A.  S.  ADAMS, WHO DIRECTED THE RESEARCH TEAM THAT
PRODUCED THF NEW ASSESSMENT OF THE THORIUM RESFRVE, SAID IN A TELEPHONE INTERVIEW THAT THF ESTIMATF WAS 'PROBABLY VERY C
ONSERVATIVE.' STOCKS EDGE OFF AS TRADING EASES - SHORT COVERING AND PROFIT TAKING CONTRIBUTE TO DROP - AVFRAGE FALLS BY
1.71 - TURNOVER IS 4,571,460 - LOSS ELICITS LITTLE SURPRISE ON WALL STRFET - GAINS TOP DECLINES,563-498 - BY RICHARD RUT
TER - THE STOCK MARKET HAD A MILD DECLINE YESTERDAY AS TRADING TAPERED OFF.  THE MINOR SET - BACK WAS NEITHER SURPRISING
 NOR ALARMING TO MARKFT FOLLOWERS AND OBSERVERS.  IT FOLLOWED NOT ONLY THREE DAYS OF GAINS BUT A GENERAL ADVANCE THAT HA
S PERSISTED FOR THE BFTTER PART OF THE LAST FIVE WEEKS.  IT WAS AN ADVANCE THAT SAW STOCKS, ON AVERAGE, POST THE BEST MO
NTHLY GAINS SINCE THE NEW YORK TIMES BEGAN TO COMPILE MARKET AVERAGES IN 1911.  VOLUMF FOR NOVEMBER WAS THE LARGEST FOR
THE MONTH SINCE 1928.  YESTERDAY, BOTH PROFIT TAKING AND SHORT COVERING WAS EVIDENT.  STOCKS OPENED LOWER ON THE NEW YOR
K STOCK EXCHANGE, DRIFTED DOWN UNTIL ABOUT NOON AND THEN FIRMED.  THE LATE RECOVERY WAS NOT QUITE SUBSTANTIAL ENOUGH TO
ERASE ALL THE EARLIER LOSSFS, BUT VARIOUS OIL, RAIL AND SPECIAL SITUATION ISSUES HAD GOOD GAINS.  THE LATFST BATCH OF EC
ONOMIC NEWS WAS ENCOURAGING, PERHAPS ENOUGH TO AVERT A LARGER LOSS.  CONSTRUCTION CONTRACT AWARDS IN OCTOBER WERE REPORT
ED AT A RECORD, PRESAGING A CONTINUED BUILDING BOOM.  MANUFACTURES NEW ORDERS ALSO REACHED A NEW HIGH, ALTHOUGH SALES DI
PPED.  AUTO PRODUCTION WAS AT THE HIGHEST LEVEL IN ALMOST THREE YEARS.  THE FACT THAT VOLUME CONTRACTED ON THE MARKET DI
P WAS ALSO TAKEN AS A MORE OR LESS BULLISH DEVFLOPMENT.  VOLUME ON THF NEW YORK STOCK FXCHANGE WAS 4,571,460 SHARES, DOW
N FROM 5,810,000 ON THURSDAY.  IT WAS THE FIRST TIME IN THE LAST SEVEN TRADING SESSIONS THAT TURNOVER HAD FALLEN BELOW T
HE 5,000,000- SHARE LEVEL.  UTILITIES TO LEAD IN OFFERINGS OF BONDS DURING COMING WEEK - NEW ISSUES OF FIXED - INTEREST
SENIOR SECURITIES EXPFCTED NEXT WEEK WILL AGAIN BE LED BY UTILITY OFFERINGS.  TOTAL MUNICIPAL AND CORPORATE DEBT SECURIT
IES SCHEDULED FOR SALF EXCEED $186,000,000, AGAINST $224,000,000 THIS WEEK.  BOTH FIGURES ARE BASED ON ISSUES OF $1,000,
000 OR MORE.  THE TOTAL OF CORPORATE AND MUNICIPAL DEBT ISSUES EXPECTED IN DECEMBER WAS PUT AT $647,000,000, COMPARED WI
TH ABOUT $693,000,000 IN THE 1961 PERIOD, ACCORDING TO THE COMMERCIAL AND FINANCIAL CHRONICLE, A TRADE PUBLICATION.  OF
NEXT WEEKS SCHEDULED OFFERINGS, THE LARGEST COMPETITIVE SALE IS $60,000,000 OF CONSOLIDATED EDISON COMPANY OF NEW YORK B
ONDS, WHICH IS EXPECTFD ON WEDNESDAY.  MAJOR OFFERINGS EXPECTED MONDAY INCLUDE $15,000,000 OF METROPOLITAN EDISON COMPAN
Y BONDS; $9,000,000 OF FLORIDA DEVELOPMENT COMMISSION REVENUE BONDS $9,900,000 OF GARY, IND., SANITATION DISTRICT BONDS,
 AND $8,000,000 OF DALLAS COUNTY, TEX., BONDS.
```

Figure 13B   OUTPUT TEXT OF NEW YORK TIMES ARTICLES

attempt was made to arrange the receiver output copy in columnar format.

The results obtained by processing the four samples through the system are tabulated in Table 15.

Sample 1 is the short message on "speed reading" which was shown in Figure 5. A compression of 63.50 percent was achieved and the output digits per character were 2.20. The latter value is close to the 1.96 binary digits per character achieved in the encoding of the dictionary. The sample was chosen to ensure that no words would require enciphering in the letter mode. More than 85 percent of the input words were directly located in the dictionary and less than 15 percent were synthesized utilizing dictionary stems.

Sample 2 is the New York Times article previously illustrated. The compression achieved was 51.90 percent with 2.89 binary digits per input character. This represents a saving of 31,765 binary digits from a total of 61,206 binary digits required in the input message (calculated on the basis of 6-channel tape). Of the 1,635 word tokens, 73.21 percent were found in the dictionary and 14.62 percent were synthesized using dictionary stems. Only 12.17 percent of the tokens were spelled out.

Sample 3A was similar to sample 2 and contained 1,593 word tokens excerpted from the Wall Street Journal. The compression ratio and output digits per character are nearly identical with those of sample 2.

A more extensive sample was tested in the 19,710 words of sample 4 that were obtained from seven current articles from four magazines. A compression of 46.8 percent was achieved with the saving of 320,872 binary digits from a total of 685,140 binary digits required in the input message (calculated on the basis of 6-channel tape). The output digits per character

TABLE 15

COMPRESSION AND INFORMATION CONTENT OF

PROCESSED MESSAGES

NO.	MESSAGE SOURCE	TOKENS	INPUT CHARACTERS	OUTPUT DIGITS	OUTPUT DIGITS/ CHARACTER	PERCENT COMPRESSION
1	Speed Reading	97	562	1,231	2.20	63.50
2	New York Times	1,635	10,201	29,441	2.89	51.90
3A	Wall St. Journal	1,593	10,109	29,153	2.88	51.94
3B	Wall St. Journal	1,593	10,109	34,943	3.45	42.38
4	Magazines	19,710	114,190	364,268	3.19	46.84

NO.	MESSAGE SOURCE	PERCENT WORD TOKENS		
		IN DICTIONARY	SYNTHESIZED	SPELLED
1	Speed Reading	85.57	14.43	-
2	New York Times	73.21	14.62	12.17
3A	Wall St. Journal	70.81	15.87	13.32
3B	Wall St. Journal	70.81	-	29.19
4	Magazines	79.49	12.18	8.33

rose to 3. 19.

Analysis reveals that of the 4, 133 word types contained in the 19, 710 words, 1, 941 were found in the dictionary, 1, 236 were synthesized from stems and suffixes found in the dictionary, and 956 were new words that were spelled out.

Of the word tokens, 15, 777 were found in the dictionary, 2, 335 were synthesized, and 1, 598 were spelled out. The spelled words comprised only 8. 1 percent of the total words. With the synthesized words, over 91 percent of the words in the input text utilized the dictionary.

B. Effectiveness of Word Synthesis Program

The effectiveness of the word synthesis program was studied in samples 3 and 4. In sample 4, containing 19, 710 word tokens, the output binary digits per input character were computed in the three word categories. The 15, 777 word tokens encoded in the dictionary had an average information content of 2. 76 binary digits per character as compared with the 1. 96 value of the dictionary. The 1, 236 synthesized tokens had an average information content of 3. 16 binary digits per character.

For the spelled words, 4. 71 binary digits per input character were required exclusive of the symbols used to switch to letter mode and return to word mode. Including these latter symbols, a total of 6. 37 binary digits per input character was required. The latter value is greater than the 6 binary digits required in the input per character. It is now known that the letter mode symbol should be the most frequent symbol in the dictionary and hence should have the shortest code. Nearly 5, 000 binary digits were lost in the mismatch of the assigned code and the actual frequency of this symbol.

The effectiveness of the synthesis routine is shown by the slight increase, from 2.76 to 3.16 binary digits per input character, in the average number of binary digits used in transmitting synthesized words. It was possible to save nearly 5,000 computer words (1,236 times four computer words) in extra storage at the cost of 0.40 binary digits per character. On the other hand, a spelled word required 6.37 binary digits per character. Thus a word can be synthesized at a little more than half the cost of spelling it out.

To further test the effectiveness of the word synthesis program, sample 3A was reprocessed with the word synthesis subroutine bypassed in the computer program. The results are summarized in Table 15 under message 3B. The compression ratio fell from 51.94 to 42.38 and the information content rose from 2.88 to 3.45 binary digits per character. Nearly 5,800 additional binary digits were required to transmit the same information.

The average length of the synthesized words in the above sample was 7.23 characters per word (excluding the terminal space) compared to 3.95 characters per word for the 5,153 words in the dictionary. Since a letter mode code and a terminal space are required by the spell out routine, these added digits plus the larger average number of characters per word which characterize words not listed in the dictionary make the synthesis routine a desirable feature. The gain in compression from 42.38 percent to 51.94 percent in the small sample indicates the greater efficiency of the split format which utilizes the synthesis routine.

The word analysis program which is illustrated in Section VI provides a clear picture of the operation of the enciphering program and shows the effect of the synthesis and letter mode routines on transmission efficiency. This is demonstrated below in a portion of the message taken from the first paragraph of Figure 12A. The message is shown as it was processed for enciphering by the computer.

THE NEW YORK TIME$\frown$S NEW YORK, MONDAY, DECEMBER 3, 1962.
% U△. % S△. SEE$\frown$S % R· E· S· P· I· T· E· △IN WORLD TENSION
AS TALK$\frown$S PROCEED - CALM IS EXPECT$\frown$ED DURING LONG
% P· A· R· L· E· Y· S· △WITH SOVIET ON POST - % C· U· B· A· △
ISSUE$\frown$S - WEST EXAMINE$\frown$S STAND - % K· E· N· N· E· D· Y· △,
% M· A· C· M· I· L· L· A· N· △AND % D· E· △ % G· A· U· L· L· E· △
TO % R· E· D· E· F· I· N· E· △ BERLIN AND ARM$\frown$S VIEW$\frown$S BY
% M· A· X· △ % F· R· A· N· K· E· L· △ SPECIAL TO THE NEW YORK
TIME$\frown$S -

(Circumflex denotes synthesis; dots between letters indicate word is spelled out in letter mode. The letter mode is preceded by % which switches the program to letter mode and is followed by △, space, which returns the program to word mode.)

C. Translation Accuracy

Three types of errors can be encountered in the transmission system: 1) in the translation of input text into the enciphered digit string, 2) mutilation of the enciphered message in transmission, 3) in the translation of the received text in the process of deciphering and decoding. Error types 1 and 3 will be discussed in this section and transmission errors will be discussed in Section VIII.

The possibility of error types 1 and 3 was reviewed in [1:109-113]. It was shown that errors can arise from the creation of "idiot" words and from operation of the word synthesis program.

An "idiot" word is a legal English word created by an illiterate computer in violation of grammatical, syntactical, and etymological rules. In most cases, this type of error will be a positive error and will improve the efficiency of the transmission. Thus to form the word SHOULDER by synthesizing the comparative degree of the word SHOULD is a grammatical error but effective enciphering provided that the number of binary digits required to transmit SHOULD and the suffix ER is less than that required to spell out S·H·O·U·L·D·E·R. The previous section has shown that this condition will usually hold.

This type of positive error has been encountered in the synthesis of the following words all of which were synthesized correctly.

LAME+NT	=	LAMENT
CONVERS~~E~~+ION	=	CONVERSION
FLOUR+ISH	=	FLOURISH
REST+ATE	=	RESTATE
ANGEL+ES	=	(LOS) ANGELES
DON+ER	=	DONNER (NAME)
AND+Y	=	ANDY (NAME)
TENS~~E~~+OR	=	TENSOR

An error of synthesis which may reduce the efficiency of transmission but does not result in an output error arises when the transmitter word decomposition program appears to have two alternate choices and the program selects the first one it encounters. If NOT and NOTE were both in the dictionary, for example, and NOTABLE wasn't, since NOT NOTABLE, it will be located before NOTE which is greater than NOTABLE. Hence, decomposition and synthesis will be of the form NOT+ABLE = NOTABLE. If the code for NOT is longer than the code for NOTE to which NOTABLE is grammatically related, then a penalty is incurred by transmitting extra digits.

Many faulty constructions are precluded by the association of a SYNTAG with each word. The SYNTAG allows certain constructions to be made and prohibits others. RAG (SYNTAG 2), for example allows formation of the word RAGGED but synthesis of RAGED is prohibited. The latter word can only be synthesized from RAGE (SYNTAG 1). When either RAG or RAGE appear alone in the dictionary, an error may arise in the output because both RAGGED and RAGED will be decomposed into RAG+ED and the output will depend on which SYNTAG is associated with the word stem.

Only three errors of this type were encountered in the 19,710 word sample, all arising from SYNTAG 2 which calls for doubling the final consonant of a word when affixing a suffix beginning with a vowel. The errors involved, at the same time, construction of idiot words. Two of these words were proper names. Thus JAM+ES was translated into JAMMES at the receiver and PET+ER into PETTER. Similarly, POT+ION was printed out as POTTION.

One exception to the SYNTAG 1 rule was encountered: ADVANTAG$\cancel{E}$+OUS = ADVANTAGOUS, the final E being elided.

The simplest way to handle this problem is to either include the problem words in the dictionary or include both SYNTAG 1 and SYNTAG 2 stems to avoid confusion.

The space routine is highly efficient and as noted in Figure 13, although a few typographical errors appear, none involved loss of information. A larger symbol set and minor changes in the punctuation routine designed to remove all ambiguities in spacing will remedy this problem.

VI. DICTIONARY ADAPTATION

In this section the problem of adapting the dictionary to usage is investigated. It has previously been shown that the maximum efficiency of transmission is achieved when an encoding matches the statistics of an information source. An optimum encoding has been defined as one which yields the lowest possible average message length for a given number of channel symbols and for a message ensemble consisting of a finite number of messages. In an operating transmission system it becomes necessary to express the finite number of messages as a specific number so that the average message length can be determined.

If transmission begins in letter mode and a dictionary is grown, it is required, after processing a stated number of word tokens, to encode the dictionary. If system operation begins with a priori statistics, it is required to compare the a priori with the a posteriori statistics at stated intervals to ensure that the encoding continues to match the a posteriori statistics. In either case, procedures for determining the match between an encoding and the statistics of the information source are required. It is also necessary to formulate criteria by which a mismatch can be detected or, conversely, when a proper match has been obtained.

If a finite set of words is assumed consisting, for example, of those words contained in a standard dictionary, the processing of a large enough number of messages will eventually entail use of all these words. A curve of word types plotted against cumulative word tokens will not be a linear function but rather an exponential function whose maximum value will not exceed the number of word types contained in the standard dictionary.

Because it has already been noted that from 500 to 1000 word types comprise approximately 75 percent of the word tokens in normal English language texts, and these words include the basic, common "function" words, the accretion curve of new word types as a function of cumulative word tokens will rise sharply at the origin and will approach the asymptote. The curve will be of the form

$$h = A - Be^{-CN}$$

where N is the number of word types in the standard dictionary and A, B, and C are parameters to be empirically determined.

An accretion curve obtained from analysis of the 19,710 word text previously described is shown in Figure 14. Curves for the three categories of words transmitted, dictionary, synthesized, and spelled are shown separately together with a curve showing total word types.

By restricting the size of the dictionary, as in a microglossary which contains a subset of the language, it can be expected that the accretion curve will approach its asymptote more rapidly. Figure 15 is the accretion curve of a 50,000 word sample excerpted from the Proceedings International Conference on Scientific Information (ICSI), Washington, 1959.* The selected papers deal with information processing and hence approximate a microglossary.

Both curves exhibit the sharp rise that was predicted and the exponential characteristic although sufficient data were not obtained to extend the curves. It is interesting to note that only 1,570 dictionary word types were found in the 50,000 word tokens of the ICSI sample whereas 1,941

*Excerpts are from following pages of the published Proceedings: 461-474, 613-645, 889-901, 903-915, 1221-1243, 1327-1363.

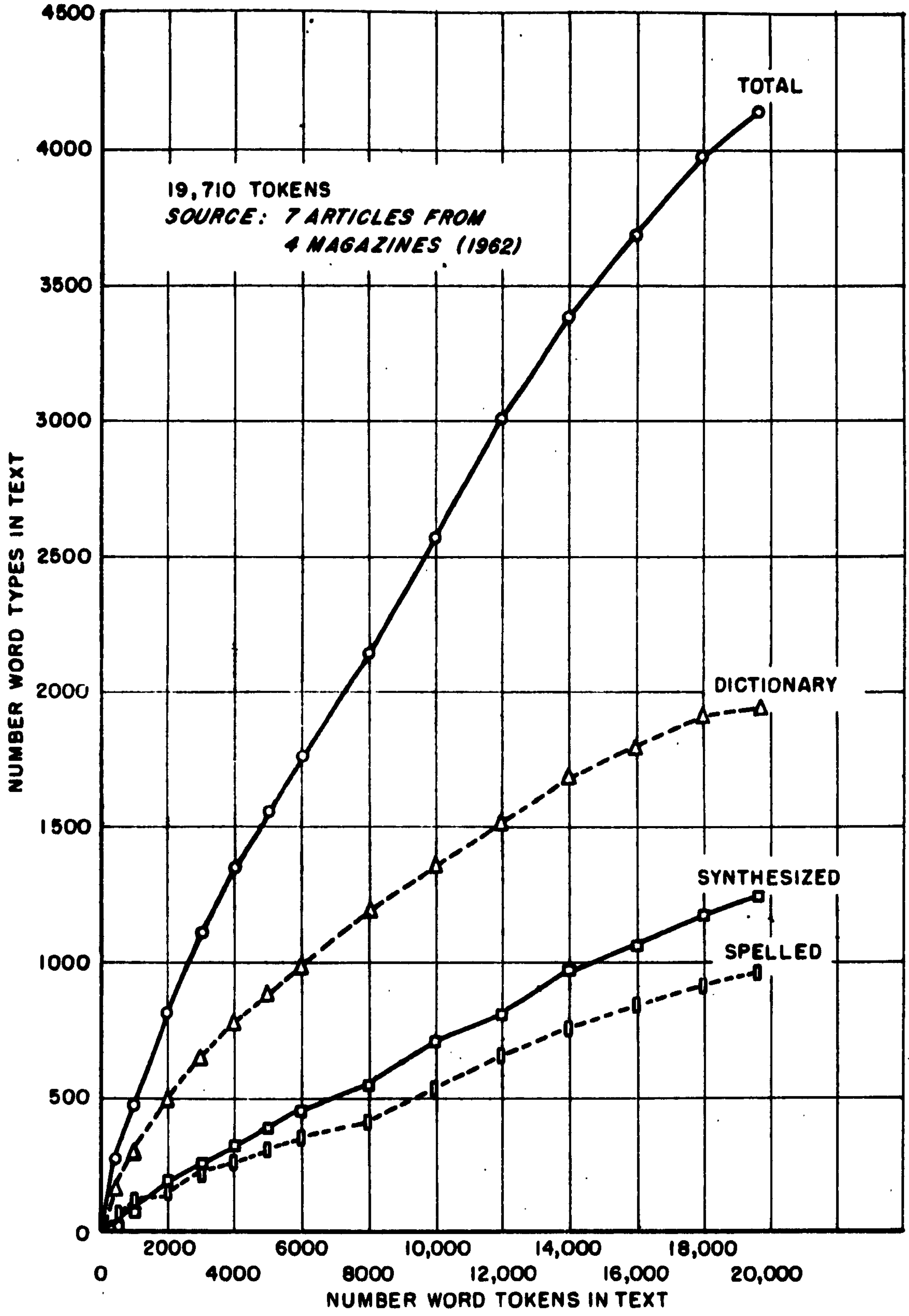


FIG. 14 ACCRETION OF WORDS IN GROWING DICTIONARY

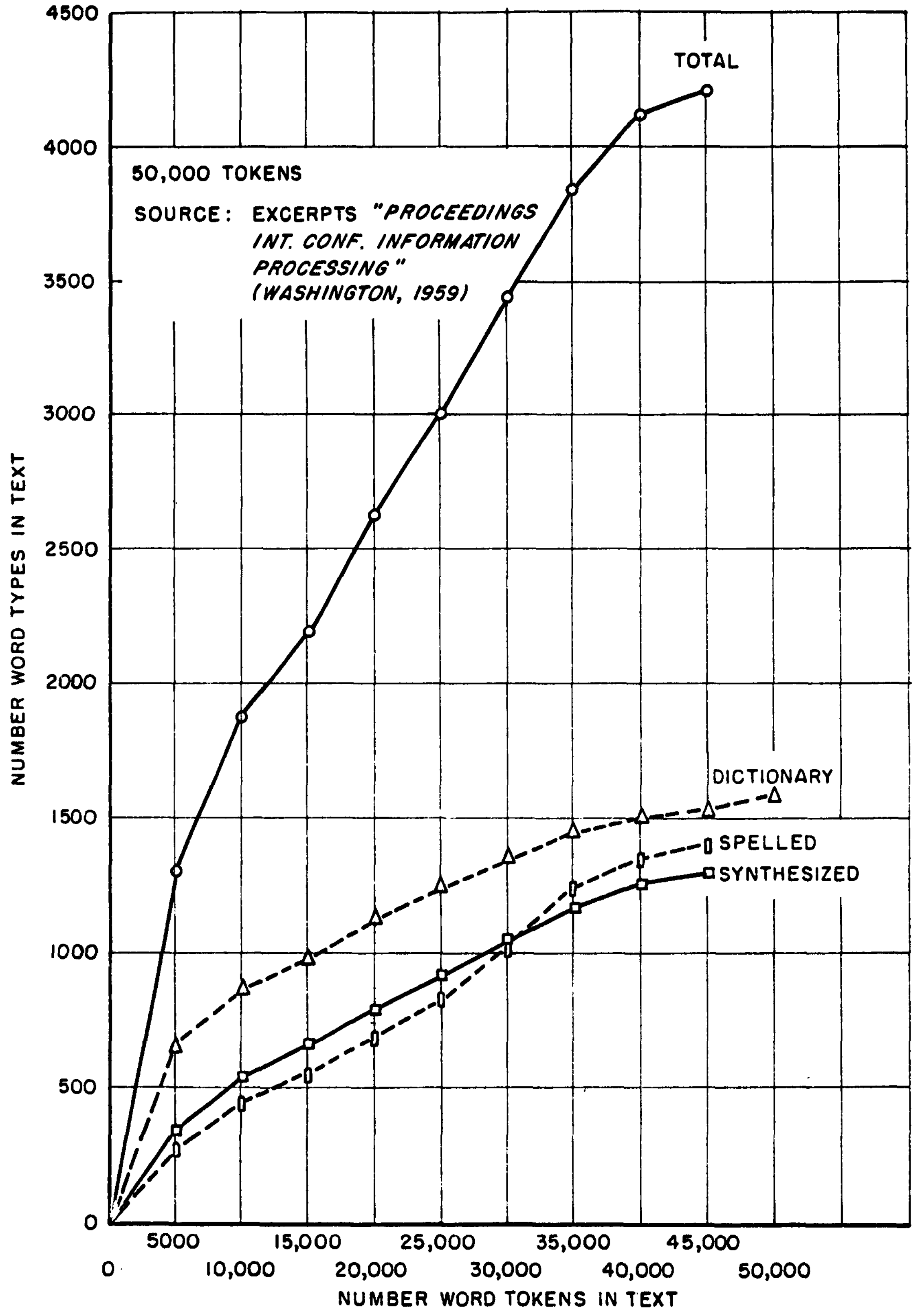


FIG. 15 ACCRETION OF WORDS IN MICROGLOSSARY

dictionary types were found in the 19, 710 word sample. The ICSI text made use of fewer common English words while the magazine texts used a more general vocabulary drawn from the 5, 153 most frequently used types.

It is also noted that at 30, 000 word tokens, the ICSI curves of synthesized and spelled words meet and the accretion of further words shows that the number of spelled word types exceeds the synthesized word types and approaches the dictionary word types. This can be explained by the presence of many specialized words in the ICSI Proceedings that are not in common use and hence were not included in the system dictionary. However, most of the spelled words appear with low frequency as evidenced by the division of word tokens by category at the 50, 000 word token point: 73. 6 percent of the tokens appear in the dictionary, 13. 1 percent are synthesized, and 13. 2 percent are spelled out.

(Approximately one percent of the words in the ICSI sample had been recorded on magnetic tape with errors. The bulk of the errors arose from words run together, that is, a space was omitted between two word types. Accordingly, the run-together words were treated by the computer as one word and spelled out since they were not located in the dictionary. These errors have been subtracted from the spelled word category where they produced the greatest distortion and have been eliminated from the counts. It is estimated that most of the error words would fall into the dictionary or synthesized categories and that with 50, 000 word tokens, most of the word types would already have appeared at least once. The resultant errors in the curves are, therefore, considered to be negligible. Spelling out all the errors letter-by-letter, the system operated with a compression ratio of 53. 7 for the 50, 000 word sample and an information content of 2. 78

binary digits per input character.)

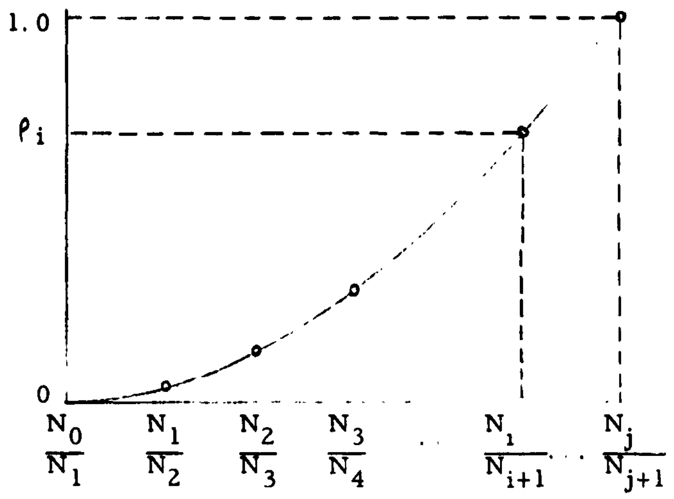
Given a complete accretion curve, or its equation, it would be possible to determine at what value of token words a designated value of word types would be found. Since this value of word types would include all of the most frequently occurring words, it is possible to establish some point on the accretion curve which, when reached, would call for an encoding that would afford efficient transmission. The accretion of additional words to the total would be expected to have little significant effect on the relative frequencies of the large majority of the word types.

A more direct way of accomplishing this same objective is through use of rank correlation methods. Consider, for example, Spearman's coefficient:

$$\rho = 1 - \frac{6\sum d^2}{n^3 - n}$$

where $\sum d^2$ is the sum of the squares of the rank differences and n is the number of word types ranked.

To compute this coefficient, let us mark off the abscissa of the accretion curve into equal increments of N word tokens, beginning at the origin with N_0. The rank correlation of words appearing at two successive increments can then be calculated. Thus ρ_1 would be equal to the rank correlation of the word list at N_0 and N_1, ρ_2 would equal the rank correlation at N_1 and N_2, etc. Since the maximum value of $\rho = 1$, a curve would be obtained as shown below.



$(N_{j+1} - N_j = \text{constant}, \ j = 0, 1, 2, \ldots)$

At a given point on the rank correlation curve a value of ρ_i would be reached. Word processing would continue until this value of ρ_i was reached rather than according to pre assigned values of N_i and N_{i+1}. The desired value of ρ_1 will have to be empirically determined as that point on the rank correlation curve at which a further increment of word tokens will produce no significant change in the rank of the word types. The word list will be said to have converged making possible an optimum encoding at this value of ρ_i.

Since thousands of word types can appear in the word lists, it is of interest to consider the computational aspects of obtaining the rank correlation coefficient. The canonical encoding of two word lists containing equal numbers of words and with the same rank-frequency characteristics will be essentially the same although the codes assigned to particular words can differ. Given the Zipf-Mandelbrot curve of Figure 7 for the experimental dictionary, for example, no change would occur in the curve if the first and second words were interchanged, or any other reshuffling of words took place.

Consider the B-sets of the canonical encoding of the experimental dictionary as tabulated on page 56. Three words are assigned codes of four binary digits, one word is assigned a code of five binary digits and six words are assigned codes of six binary digits. If THE ranks first in one list, its code will be four digits in length. If I ranks sixth, its code will be assigned six digits. In the second list, the ranks may be reversed, and I would be assigned a four-digit code and THE a six digit code.

Let us now consider that the two lists above are derived from the a priori and a posteriori statistics of an information source. Since I appears very frequently and its code is mismatched with its a posteriori rank, every transmission of the word I will penalize the transmission system by sending superfluous digits. Hence, it is important to compare the ranks of the most frequent words in the word lists to ensure that a minimum redundancy encoding has been obtained for the a posteriori ranks

Now examine both the B-set table on page 56 and Table 4 on page 28 which lists the frequency distribution of the 5, 153 most frequent words in the experimental dictionary. From the B-set table it is noted that 591 words have codes of length 12 binary digits or less. These 591 codes make up approximately 80 percent of the word tokens in the 4. 26 million word sample. The next B-set contains 519 words of length 13 binary digits. If a word in the a priori list is ranked 600 and in the a posteriori list is ranked 900, no change in code length will be affected by the discrepancy in rank of the two lists.

Hence the possibility arises of computing the correlation coefficient with only the first several hundred words on the N_i and N_{i+1} word lists and disregarding the ranks of the bulk of the word types that comprise but a small percentage of the total word tokens. It remains to determine the error that will be introduced in the value of ρ by truncating the word lists.

It is hypothesized, therefore, that by computing the rank correlation coefficient between word types obtained at two points on the accretion curve of a growing dictionary, considering only that fraction of word types that comprise the bulk of the word tokens, it can be determined when an encoding derived from the last word list will have converged to make an optimum encoding possible. The optimum encoding will, of course, be dependent upon a designated value of the rank correlation coefficient.

Although sufficient data have not been obtained to verify the assumptions and hypotheses presented in the previous pages, the experimental system has been designed to make these tests. Computer programs have been written to collect these data and to adapt a growing dictionary to usage.

The analysis of input text for growing and adapting a dictionary is made using a word analyzer program which sorts and counts words appearing on the "new word" tape and merges them with the counted dictionary words. A flow chart of these procedures is shown in Chart 6 of Section IX.

Total word and character counts together with counts of individual letters, symbols, and suffixes for word sample 2 of Section V, are shown in Table 16. A partial list of the most frequent words contained in the input text that were found in the dictionary is shown in Table 17. A partial list of synthesized words as processed by the word analyzer computer program is shown in Table 18. A similar list of the spelled words appears in Table 19. Lists of the dictionary, synthesized, and spelled words were merged and ordered according to frequency. The most frequent words in this list are shown in Table 20. A D indicates dictionary word, S is a synthesized word, and N is a non-synthesized or spelled word. Figures for number of letters and output bits are per word. The computer program also prints out the cumulative sums for total number of letters and binary digits in each category. These figures have been given in the previous summary.

A merged and sorted list as illustrated in Table 20 is the basis of the adaptive system. Given a list prepared at the i-th interval and another at the (i+1)-st interval, the rank correlation between the two lists can be computed.

The 60 most frequent word types obtained after processing 10,000 and 19,710 word tokens of sample 4 are shown in Table 21. The word, frequency, and category are printed from computer output. The rank numbers have been inserted.

It is noted that the ranks of the first seven words in the two lists are identical and ranks 8-9-10 in the 19,710 word list appear as ranks 9-10-8 in the 10,000 word list. The rank correlation between the top ten words of the lists is 0.96. The ten words comprise 23 percent of the word tokens in the 19,710 word sample. The rank correlation between the top 20 words drops to 0.41 at which point 30 percent of the tokens of the larger sample have been included. A comparison of the top 30 words with 35 percent of the tokens of the larger sample represented yields a rank correlation coefficient of 0.28.

10201	Total No. Input Characters
1635	Total No. Words
1197	No. Words in Dictionary
239	No. Words Synthesized
29441	Total No. Output Bits
199	No. Words Spelled Out
1731	Total Input Spaces

SYMBOL FREQUENCIES

Space	32
-	47
0	106
1	26
2	10
3	4
4	8
5	10
6	11
7	5
8	[illegible]
9	12
'	2
,	112
.	93
!	2
"	11

LETTER FREQUENCIES

A	621
B	131
C	249
D	319
E	903
F	172
G	136
H	361
I	634
J	11
K	66
L	321
M	204
N	570
O	621
P	153
Q	7
R	483
S	570
T	767
U	213
V	86
W	158
X	23
Y	114
Z	7

SUFFIX FREQUENCIES

AL	2
ATION	1
D	17
ED	35
ER	3
ES	14
EST	1
IC	1
ING	25
ION	2
ISH	1
LY	4
N	5
R	2
S	123
ST	3

TABLE 16. TOTAL WORD AND CHARACTER COUNTS.

DICTIONARY WORDS

WORD	FREQUENCY	LETTERS
THE	134 0	3
OF	65 0	2
TO	53 0	2
IN	34 0	2
A	29 0	1
AND	27 0	3
NEW	21 0	3
THAT	19 0	4
WAS	19 0	3
IS	18 0	2
WITH	15 0	4
BY	14 0	2
ON	14 0	2
FOR	12 0	3
IT	11 0	2
WERE	10 0	4
YORK	10 0	4
AS	9 0	2
HAVE	9 0	4
AT	8 0	2
BE	8 0	2
HAS	8 0	3
THIS	8 0	4
WILL	8 0	4
BEEN	7 0	4
BUT	7 0	3
FROM	7 0	4
NOT	7 0	3
OIL	6 0	3
RADIO	6 0	5
SAID	6 0	4
AN	5 0	2
CHINESE	5 0	7
DURING	5 0	6
LAST	5 0	4
MR	5 0	2
OFF	5 0	3
OR	5 0	2
RESERVE	5 0	7
SPECIAL	5 0	7
THEIR	5 0	5
UNION	5 0	5
WEEK	5 0	4
ABOUT	4 0	5
CHRISTIAN	4 0	9
GOVERNMENT	4 0	10
HAD	4 0	3
MARKET	4 0	6
SOVIET	4 0	6
THEY	4 0	4
UNITED	4 0	6

TABLE 17. DICTIONARY WORDS FREQUENCY LIST.

SYNTHESIZED WORDS

WORD	FREQUENCY	LETTERS	OUTPUT BITS
TIMES	7 S	5	18
TALKS	6 S	5	20
BONDS	6 S	5	23
EXPECTED	5 S	8	21
ISSUES	5 S	6	23
INDIAN	5 S	6	30
REPORTED	4 S	8	22
GAINS	4 S	5	23
OFFERINGS	4 S	9	25
PROBLEMS	3 S	8	21
TROOPS	3 S	6	24
DEMOCRATS	3 S	9	25
LUCKES	3 S	6	25
SCHEDULED	3 S	9	27
ESTIMATES	3 S	9	24
STOCKS	3 S	6	22
TRADING	3 S	7	22
ARMS	2 S	4	20
EXPECTS	2 S	7	21
RUSSIANS	2 S	8	24
OFFICIALS	2 S	9	24
FLANKING	2 S	8	25
TRAPPED	2 S	7	23
APPROACHED	2 S	10	22
HOUSING	2 S	7	19
MEETINGS	2 S	8	22
MILES -	2 S	5	21
DETERMINED	2 S	10	26
COMPANIES	2 S	9	24
TONS	2 S	4	25
EARLIER	2 S	7	22
COVERING	2 S	8	21
TAKING	2 S	6	19
WEEKS	2 S	5	20
LARGEST	2 S	7	25
SECURITIES	2 S	10	25
SEES	1 S	4	19
EXAMINES	1 S	8	24
VIEWS	1 S	5	22
LOOKING	1 S	7	18
DIPLOMATIC	1 S	10	30
DISCUSSIONS	1 S	11	24
TENSIONS	1 S	8	25
MONTHS	1 S	6	20
RELAXATION	1 S	10	26
RESOLVED	1 S	8	28
REMOVAL	1 S	7	24
POSSIBILITIES	1 S	13	25
INFLUENCED	1 S	10	26
RELATIONS	1 S	9	23
BELIEVED	1 S	8	24

TABLE 18. SYNTHESIZED WORDS FREQUENCY LIST.

SPELLED WORDS

WORD	FREQUENCY	LETTERS	OUTPUT BITS
SOCIALISTS	6 N	10	46
THORIUM	.5 N	7	33
DR	5 N	2	9
PARLIAMENT	5 N	10	44
CUBA	4 N	4	23
DEC	4 N	3	14
ADENAUER	4 N	8	33
S	3 N	1	5
RESPITE	3 N	7	29
NUCLEAR	3 N	7	32
NORTHEAST	3 N	9	37
COALITION	3 N	9	39
LUCKE	3 N	5	28
LIBYA	3 N	5	27
REIUA	3 N	5	23
USING	3 N	5	26
C	2 N	1	6
PARLEYS	2 N	7	34
REDEFINE	2 N	8	32
ADMINISTRATION	2 N	14	60
CUBAN	2 N	5	27
NOV	2 N	3	15
CHANCELLOR	2 N	10	46
ADENAUERS	2 N	9	38
DEPUTIES	2 N	8	36
BENGHAZI	2 N	8	45
TORIS	2 N	5	22
PIPELINES	2 N	9	40
HAMPSHIRE	2 N	9	42
JOHN	2 N	4	23
ASSESSMENT	2 N	10	44
TURNOVER	2 N	8	36
CORPORATE	2 N	9	39
EDISON	2 N	6	25
KENNEDY	1 N	7	34
MACMILLAN	1 N	9	44
DE	1 N	2	8
GAULLE	1 N	6	30
MAY	1 N	3	19
FRANKEL	1 N	7	34
INCONCLUSIVE	1 N	12	58
EXPLORATORY	1 N	11	54
INTERNATIONAL	1 N	13	52
ACCOMMODATION	1 N	13	61
JET	1 N	3	17
BOMBERS	1 N	7	36
IMMINENT	1 N	8	35
NEGOTIATION	1 N	11	46
CASTRO	1 N	6	27
REGIME	1 N	6	27
MOSCOWS	1 N	7	37

TABLE 19. SPELLED WORDS FREQUENCY LIST.

MERGED AND SORTED WORDS

WORD	FREQUENCY	LETTERS
THE	134 D	3
OF	65 D	2
TO	53 D	2
IN	34 D	2
A	29 D	1
AND	27 D	3
NEW	21 D	3
THAT	19 D	4
WAS	19 D	3
IS	18 D	2
WITH	15 D	4
BY	14 D	2
ON	14 D	2
FOR	12 D	3
IT	11 D	2
WERE	10 D	4
YORK	10 D	4
AS	9 D	2
HAVE	9 D	4
AT	8 D	2
BE	8 D	2
HAS	8 D	3
THIS	8 D	4
WILL	8 D	4
TIMES	7 S	5
BEEN	7 D	4
BUT	7 D	3
FROM	7 D	4
NOT	7 D	3
TALKS	6 S	5
BONUS	6 S	5
SOCIALISTS	6 N	10
THORIUM	6 N	7
OIL	6 D	3
RADIO	6 D	5
SAID	6 D	4
EXPECTED	5 S	8
ISSUES	5 S	6
INDIAN	5 S	6
DR	5 N	2
PARLIAMENT	5 N	10
AN	5 D	2
CHINESE	5 D	7
DURING	5 D	6
LAST	5 D	4
MR	5 D	2
OFF	5 D	3
OR	5 D	2
RESERVE	5 D	7
SPECIAL	5 D	7
THEIR	5 D	5

TABLE 20. COMPOSITE LIST OF MOST FREQUENT WORDS OF ALL CATEGORIES.

TABLE 21

RANK-FREQUENCY ORDER AT TWO POINTS ON ACCRETION CURVE

10,000 TOKENS				19,710 TOKENS		
RANK	WORD TYPE	FREQUENCY		RANK	WORD TYPE	FREQUENCY
1	THE	557 U		1	THE	1192 U
2	OF	324 U		2	OF	677 U
3	A	274 U		3	A	541 U
4	TO	247 U		4	TO	518 U
5	AND	234 U		5	AND	462 U
6	IN	196 U		6	IN	450 U
7	THAT	109 U		7	THAT	242 U
8	IT	105 U		8	HE	195 U
9	HE	97 U		9	IS	190 U
10	IS	97 U		10	IT	181 U
11	FOR	79 U		11	FOR	157 U
12	WE	79 U		12	HIS	139 U
13	ON	78 U		13	ON	134 U
14	I	75 U		14	ARE	124 U
15	HIS	63 U		15	BE	123 U
16	WAS	64 U		16	WITH	121 U
17	THEY	62 U		17	I	112 U
18	YOU	62 U		18	HAVE	111 U
19	WITH	61 U		19	WAS	111 U
20	AS	59 U		20	YOU	106 U
21	OUR	59 U		21	BY	106 U
22	ARE	51 U		22	THEY	104 U
23	BEAR	50 U		23	AS	103 U
24	NOT	49 U		24	WE	100 U
25	HAVE	48 U		25	OR	95 U
26	THIS	47 U		26	BUT	94 U
27	WHEN	45 U		27	THIS	91 U
28	AN	44 U		28	NOT	85 U
29	AT	44 U		29	AN	82 U
30	BE	43 U		30	WHO	81 U
31	BUT	43 U		31	AT	81 U
32	BY	40 U		32	ONE	80 U
33	FROM	39 U		33	WHEN	76 U
34	OUT	39 U		34	THEIR	71 U
35	HAD	38 U		35	FROM	71 U
36	OR	38 U		36	HAD	70 U
37	WHO	37 U		37	OUR	70 U
38	ONE	35 U		38	OUT	66 U
39	THERE	33 U		39	ALL	62 U
40	HAS	32 U		40	SCHOOL	60 U
41	WERE	32 U		41	WERE	60 U
42	HIM	31 U		42	IF	59 U
43	THEM	30 U		43	HAS	59 U
44	ALL	27 U		44	HIM	57 U
45	DO	26 U		45	WHICH	54 U
46	IF	26 U		46	THEM	52 U
47	THEIR	26 U		47	THERE	52 U
48	NO	25 U		48	UP	52 U
49	TIME	25 U		49	NO	51 U
50	WHICH	25 U		50	BEAR	50 U
51	POLAR	24 N		51	CAN	47 U
52	SHE	24 U		52	MORE	46 U
53	WHAT	24 U		53	INTO	45 U
54	BEEN	23 U		54	WHAT	45 U
55	SAID	23 U		55	BEEN	44 U
56	UP	23 U		56	HER	43 U
57	US	23 U		57	SO	43 U
58	YEARS	22 S		58	ABOUT	42 U
59	HER	22 U		59	SOME	41 U
60	LIKE	22 U		60	YEARS	40 S

A comparison was also made between the word lists after processing 1,000 tokens and the 10,000 tokens. Only eight of the ten most frequent words at the 10,000 word interval were found among the top ten at the 1,000 word interval. The rank correlation between the most frequent ten words at these two intervals was 0.04. The rank correlations between the top 20 and 30 words at these intervals had negative values.

It is thus seen that after processing an additional 10,000 words beyond the 10,000 word interval, the ranks of the ten most frequent words had become nearly identical (ρ = 0.96) and the rank correlations of the top 20 and 30 words have risen from negative values to 0.41 and 0.28, respectively. Processing additional increments of 10,000 words would continue this development and the rank correlations between the top 20, 30, 40, . . . words would increase until a value of ρ_1 was reached upon comparing those word types which comprise a designated percentage of the word tokens.

Those S and N word types of the merged and sorted list whose frequencies are high can be included in the dictionary upon re-encoding and those dictionary words with frequency less than the cut-off frequency as determined by storage considerations can be deleted. In Table 21, for example, it is noted that the word YEARS (rank 60 in 19,710 tokens) has been synthesized. If the dictionary were re-encoded, YEARS would be inserted as a word type and would be assigned a code since it is more costly to transmit YEAR+S.

VII. SYSTEM ORGANIZATION

Numerous decisions have been made in determining the system organization. The present organization is based on a modified split dictionary with storage of most of the dictionary words on magnetic drum. The flow charts for the experimental system and one oriented towards storage of the dictionary on magnetic tape are presented in Section IX. The magnetic tape system was not programmed but was considered the best alternate method with regard to processing time [1]. From a study of the operation of the present system more can be said about the organization of the system and general systems design considerations.

The transmitter enciphering program operated at a rate of 1,000 words per minute. This is a rate of 16.7 words/second or 60 msec. per word. Since only words starting with the letters A, B, or C were stored in core a considerable amount of time was used in accessing the drum. For each word found in the dictionary which was stored on the drum, two average access times were used (34 msec.). The first access obtains the group of words in which the word is stored and the second increments the word counter which is stored on the drum. The word counter is necessary in providing the adaptive feature of the dictionary. The first access also must transfer approximately 256 words from drum to core. This transfer requires 6 msec., bringing the total time to 40 msec. for each word stored on the drum. This time is completely dependent on the selection of the drum as the storage media. How this affects the average time per word is dependent on the frequency of occurrence of the words stored on the drum.

The frequency of occurrence of words from a 4.26 million word sample according to their first letter was tabulated in Table 5 (page 31).

From the table, the following data are obtained:

LETTER	WORD TYPES	WORD TOKENS
A	321	515,656
B	286	175,809
C	501	166,742
TOTAL	1,108	858,207

Thus words beginning with A, B, or C occur with a relative frequency of 20 percent and only 20 percent of word search is performed directly in core storage. Eighty percent of the words require a drum to core transfer.

If a new internal BCD code or collating sequence were defined for the alphabet based on the frequency of use according to the first letter of a word, then considerable saving in processing time can be realized. Using Table 5 again, the first five letters of highest rank of token words may be stored in approximately the same amount of core storage.

LETTER	WORD TYPES	WORD TOKENS
T	289	647,530
A	321	515,656
I	191	343,210
W	179	325,421
H	198	295,335
TOTAL	1,178	2,127,152

The relative frequency of use will be approximately 50 percent. Thus only half of the incoming words would be expected to require a drum access. This reorganization would require an increase in the Buffer Storage region described in Section IX since the letter C is not included in this group and the initial trigram CON requires 94 x 4 computer words of storage, or 376 words.

This increase in the Buffer Storage region would entail other changes in the program, mainly in the three-letter table. The format of the three-letter table is also shown in Section IX. The address range for storage is determined from two consecutive locations. By changing to two tables, one with the three-letters, and one with the address range of storage, (both the lower and upper address in one word), the dictionary words could be stored in groups on either core or drum. Search would begin with the three-letter table. If the three letters are found, the associated Storage Address Table would give the address range and location. In this manner optimum use of core and drum storage can be made. With this single change in the program, it is estimated that a 30 percent increase in the speed of processing could be realized.

The receiver decipher and decoder program for the experimental system was considerably faster than the encoder: a rate of approximately 30 words/sec. was achieved with the UNIVAC 1105. The high rate was due to two factors. First, no search was involved in finding a word since as was previously explained, the input string is deciphered and the address of the word associated with a code can be determined from the code break-point table. All that is necessary is to move the word to the output area and determine spacing and punctuation from information contained in the SYNTAG. The second factor is the ordering of the dictionary words in storage. Deciphering is dependent upon ordering the dictionary according to the code length. Since the code length is in turn dependent upon the frequency of use of a word, the most frequently used words appear first in the core storage region.

The storage required is approximately the same in core and drum but the more frequently used words are stored in magnetic core storage. The number of dictionary words stored in core is 1650, the remainder being stored on drum. Using Zipf's Law, which has been shown to apply to the sample, the relative use of core and drum can be determined.

By Zipf's Law the frequency of use of a word is given by:

$$P = \frac{0.1}{r}$$

where r is the rank of the word assigned according to frequency of use. The frequency of use of the words stored in core will be given by:

$$P_T = \sum_{r=1}^{1650} \frac{0.1}{r}$$

Approximating by integration:

$$P_T = \int_{r=1}^{1650} \frac{0.1}{r}\, dr$$

$$P_T = 0.1 \ln 1650 = .7406$$

Thus 74 percent of the words processed will be found in core storage and 26 percent will be found on the drum.

Generally, the greatest single factor affecting the operation of the system is the use of core and drum storage. It is difficult to make estimates on the influence of other parts of the program such as letter mode transmission or decomposition since the relative frequency of use of these portions cannot be determined adequately. Their effect on the speed of processing will be small since the dictionary covers approximately 80 percent of all word tokens. That is, it is expected that 80 percent of all words would be found in the dictionary. The remaining 20 percent of word tokens are split approximately evenly between synthesized words, requiring decomposition, and words requiring transmission in letter mode.

Searching the dictionary is the most time consuming part of the program and any decrease in this time will be most effective. Since so few words require decomposition a more sophisticated decomposition routine may be used without seriously affecting the average processing time.

From the previous discussion it is clear that the primary consideration for using the system with another computer is the intermediate storage available. On the basis of storage alone most small scale computers are eliminated. If the computer is provided with magnetic tape handlers then it would be possible on a smaller machine to use a system which requires pre-sort and post-sort routines with dictionary storage on magnetic tape. As pointed out in Section IX this system leads to the minimum computer configuration. Even though the dictionary is stored on magnetic tape, sufficient core storage or high speed internal storage must be provided for sorting in order to obtain optimum performance. The amount of storage necessary is dependent on the speed of the magnetic tape handlers and the internal sorting speed.

In the magnetic tape system, the incoming message is broken up into words and symbols each of which is assigned a precedence number indicating its position in the message. The words in the message are then sorted alphabetically and compared to the dictionary tape, each word being replaced by its binary code. When all words of the message have been processed, the output would be re-sorted on precedence number and the sorted codes would then be transmitted. Only one pass of the dictionary tape would be required for M words in the input message. The average time per word would be

$$\frac{R + 2S}{M} \text{ seconds,}$$

where S is the internal sorting time for M words of input and R is the tape read time. The value of S will depend on the internal sorting method used. For purposes of comparison, a simple sorting method using exchanging will be considered. The time required for this sorting method is proportional to $M^2/2$. If it is assumed that the sorting time is

$$S = \frac{M^2}{2} k$$

where k is dependent on the speed of the computer, then the average sort time per word using this method would be:

$$T = \frac{R + 2\frac{M^2}{2} k}{M}$$

$$T = \frac{R}{M} + M k$$

By differentiating the above expression, the optimum number of words may be determined for minimum time. This is $M = \sqrt{\frac{R}{k}}$ and estimating R to be 8 seconds and the value of k to be 400 μ sec. (for the UNIVAC 1105), M = 200 words and T would be 120 msec. This is a rather crude estimate but it does show how the optimum number of words per input message may be determined. By this method, the average time per word would be reduced considerably but the overall delay would be in the order of R + 2S seconds from the time the message is first inserted. This method requires the least amount of core storage, requiring only M words of input in addition to storage required for the program. Because of this long overall delay the magnetic tape system was not used.

For a larger machine, such as the IBM 7090, the entire dictionary can be stored in the core storage of 32,768 computer words (36 bits per word). If access to the drum in the UNIVAC 1105 program were eliminated, the program could operate at 28 ms per word, or approximately 2,150 words per minute. Assuming an average speed ratio between the IBM 7090 and UNIVAC 1105 of 10 to 1, a program for the IBM 7090 would run at approximately 21,500 words per minute. This is an extremely fast rate but consideration must be given to the greater cost in using this machine in a system.

Only when a compromise is made between high speed storage and intermediate speed random access drum or disc storage does the system design problem become complex.

Since many of the medium to small scale computers are variable word length decimal or character machines with moderate rental rates, it is instructive to represent the storage requirements in terms of characters. Twelve characters would be required for the maximum length dictionary word as determined by this study. If a decimal machine is considered, then 24 decimal digits would be required since alphabetic information is usually encoded as two decimal digits.

The binary code associated with each dictionary word can be represented as octal digits (0-7) in a character or decimal machine. No code in the experimental dictionary exceeded 17 binary digits and thus only 6 octal digit representations would be required. For a character machine with bit manipulation commands possibly only three characters would be required.

The one word of storage provided in the experimental dictionary for the frequency counter was rather excessive since the frequency of the most frequent word in the Thorndike study of 4.5 million words, upon which the experimental dictionary was based, did not exceed 250,000. Thus providing 6 decimal digits or characters is sufficient. In the worst case, 28 characters would be required for each dictionary entry, four extra characters being provided for programming purposes. For 5,000 words 140,000 characters would be required. For a decimal machine with two digit alphabetic coding, 200,000 digits would be required.

This amount of storage can only be provided on a smaller machine by magnetic drum or disc storage. In the average monthly rental range of $5,000 to $15,000 are found computers from which a very flexible system can be designed. The computers in this price range that can be used in a system based on random access drum or disc storage are:

RCA	301
IBM	1401 (tape and disc)
IBM	1410
General Electric	225
Honeywell	400
Univac	SS 80/90
NCR	315

The random access storage available on each of these machines is over a million characters and could provide for a more extensive dictionary. All these machines also have magnetic tape units which allows consideration of either form of system organization. Even for a system based on a form of high capacity random access storage, magnetic tape units would provide convenient storage for intermediate results and the master program itself.

Consideration of these computers would probably yield the most economical system for a proposed application. The larger the computer the easier it would be to design and program a system. Warrant for using a larger computer or one of the above with more optional equipment will depend on other tasks the larger system can perform. A larger computer utilized fully in a number of applications would reduce considerably the cost per computation.

Estimating the rate of operation for a system using medium sized computers is difficult without rough programs for the individual computers, but a reasonable bound can be made based on experience with the experimental system. The encipher program for the experimental dictionary processed words at a measured rate of 1000 words per minute. The effect of drum access time on this rate has been discussed previously. If storage of the dictionary were restricted to the drum, then no matter what improvement is made in the program the average time per word can never be less than 34 milliseconds, which is two average access times to the drum. Thus the greatest possible rate with the Univac 1105 would be approximately 30 words per second or 1,800 words per minute.

With a program similar to the one described for the Univac 1105 an estimate can be made of the maximum possible rate based on the average access time for the drum or disc storage of the smaller machines mentioned above as shown in Table 23.

TABLE 22

MAXIMUM WORD RATES

Machine	Average Access Time (msec.)	Maximum Rate (Words/sec.)
RCA 301	100	5. 00
IBM 1401	600	0. 83
IBM 1410	160	3. 10
GE 225	158	3. 16
H 400	250	2. 00
SS 80/90	385	1. 30
NCR 315	200	2. 50

An approach to cost evaluation of a transmission system utilizing minimum redundancy codes is to compare its costs with those of a standard system. The cost of a large scale digital computer runs from $5 to $15 per minute, depending on the type computer used, the storage capacity, and the operation speed.

Let us assume that the cost per word using standard teletype code is $0. 02 per word. The cost per bit of an average word of 30 bits will be $0. 0007. Using the minimum redundancy encoding for a 5,000 word dictionary, a saving of 21 bits would be realized over the teletype code, amounting to $0. 015 per word. This may be related to computer cost by determining the minimum rate at which the transmitter and receiver must operate to "break even". Assuming an average computer cost of $10 per minute, the system must process words at the rate of 10/. 015, or 660 words per minute to break even. This is a rate of about 11 words per second.

In the foregoing discussion many assumptions had to be made in regards to costs. Let us express the "break even" rate in more general terms and consider the factors in the expression

$$R = \frac{C}{\left(1 - \frac{B_w^m}{B_w^o}\right) C_w^o}$$

R	= "break even" rate
C	= computer cost per unit time
C_w^o	= transmission line costs per word
B_w^o	= bits per word using standard code
B_w^m	= bits per word using minimum redundancy word codes.

For a transmission line, such as a transatlantic cable, the cost per word would be much greater, in the order of 20 to 30 cents per word. Thus the cost, C_w^o, would be greater and a much lower "break even" rate would be required. Here the computer would be used to definite advantage and a less stringent requirement would be placed on the computer.

The choice of a split or integrated dictionary format will effect B_w^m. For a split dictionary covering the same number of words as an integrated dictionary, the coding efficiency will be less, or B_w^m greater. Of course, using the stems and suffixes in the dictionary many words which would not appear in an integrated dictionary could be synthesized. Important here is the fact that words not in the dictionary must be transmitted letter by letter, thus increasing B_w^m for an integrated dictionary. Thus the storage capacity of the computer will affect not only the purchase or rental price but also the encoding efficiency.

The monthly rental charge for a computer is generally based on 176 hours per month (8 hours per day, 22 days per month). Assuming a mid-range average monthly rental of \$10,000 for the computers considered above, the cost per hour is \$56.08 or \$0.94 per minute. With similar equipment at the transmitter and receiver and a compression ratio (B_w^m/B_w^o) of 50 percent the break-even rate for an inexpensive line is found to be quite low.

$$R = \frac{2 \times 0.94}{(1-0.5)\,0.02} = \frac{0.94}{0.005} = 188 \text{ words/min.}$$

Considering the maximum word processing rates from Table 22, a rate of 1 or 2 words/sec. (60-120 words/min.) is reasonable to expect and the system would not break even with regards to cost.

Transmission related savings would be realized for long distance transmission. For standard Telex service to London or Paris the cost is approximately \$0.05 per word based on a rate of \$3.00 per minute for 60 words per minute. The break-even rate for this higher transmission cost is:

$$R = \frac{2 \times 0.94}{0.5 \times 0.05} = \frac{1.88}{0.025} = 75.2 \text{ words/min.}$$

From the considerations above the expected rate of 60-120 words per minute would indicate the system would break-even or result in a savings at the higher word processing rate.

The above is a worse case estimate but indicates that an economical system can be designed with a smaller computer. This estimate was based on storage of the dictionary completely on drum or disc. If more magnetic core storage is available, then a higher word processing rate is possible as was discussed previously, by storing the more frequently used words in core storage.

VIII. ERROR DETECTION BY SELF-SYNCHRONIZATION OF VARIABLE LENGTH NON-SPACED PREFIX ENCODINGS

A. The Error Problem

Although the non-spaced exhaustive prefix encoding is the most efficient minimum-redundancy encoding and is the simplest to encipher and decipher, its use has been restricted because of its capability to propagate errors without detection. Error propagation arises from the exhaustive property of the encoding and any infinite sequence of channel letters, even though it contains errors, can still be deciphered as a message. Since the codes are of variable length, no indication is available as to when a given code is transformed into another code in the presence of errors.

To circumvent this difficulty and yet gain the advantage of the variable-length encoding, spaced codes have been devised. Brillouin [16] employed a distinct sequence of digits to terminate each code. Neumann [17] has carried this approach a step further in developing n-definite codes whose structure ensures that error propagation is self-limiting. Bemer [18] employed prefixes which designate the number of digits in a code.

The vulnerability of the non-spaced variable-length encoding to errors arising from loss of synchronism between transmitter and receiver presents an additional problem. The previously cited spaced codes limit the synchronization delay to a maximum number of codes that may be altered before resynchronization. The comma-free encodings developed by Golomb, Gordon and Welch [19] and the prefix synchronized encodings of Gilbert [20] establish limits on the synchronization delay.

However, none of the above codes is capable of detecting the presence of an error without contextual inspection by a human observer. If a complete code were dropped, the loss of information would again be detectable only by a human observer.

This section presents an error detection system which is derived from the property of self-synchronization of non-spaced exhaustive prefix encodings. The nature of self-synchronization and the conditions that must be met for a string of digits to return to synchronization are discussed. A procedure for determining a synchronizing sequence is developed and a system is described for detecting deletion, insertion, and transposition errors within a synchronizing sequence. Experimental results obtained from a 5,208-code non-spaced encoding are given.

B. The Prefix Property and Partition of Codes

Definitions of a code C, encoding, enciphering, and a uniquely decipherable encoding appear in Section IV. B. It will be recalled that an encoding is stated to have the prefix property if no code is a prefix of any other code in the encoding. The encoding is exhaustive if it encodes two or more symbols in a uniquely decipherable manner and, for every infinite sequence of channel letters, there is some message which can be enciphered as that sequence.

A partition on the set of digits in C is defined as a collection of disjoint subsets such that their set union is C. Every digit of C will be contained in one and only one subset of the partition. Let us restrict the number of subsets to two. The initial subset, designated α, is called a prefix. The second subset, designated β, is called the terminator. Hence, every code is of the form $C_i = \alpha \cdot \beta$ where the symbol $\cdot$ denotes concatenation. Either α or β may be the null sequence $\emptyset$. The prefix is called proper if the terminator $\beta \neq 0$. All codes of a prefix encoding are improper prefixes for they cannot form the prefix of any other code in the encoding.

Excluding the case where either α or β is the null sequence, there are n-1 ways to partition a code of n digits into two subsets. The code 1101, for example, can be partitioned into two subsets in three ways: $1 \cdot 101$, $11 \cdot 01$, $110 \cdot 1$. The prefix in each partition is a proper prefix.

A set of codes whose prefix is α_i form a prefix set. The prefix sets of an encoding form a hierarchy of sets in accordance with the method of partition. Consider, for example, the prefix sets obtained by the partition of the eight codes of Figure 16.

The α_0-set consists of codes 00 and 01, each code being partitioned into two subsets of one digit each. The α_1-set consists of the remaining six codes, each partitioned into a one digit prefix and a two or three digit terminator. The α_1-set can be divided into two subsets, α_{10} and α_{11}, by placing two digits of each member of the set in the prefix and the remaining digits in the terminator. Similarly, the α_{11}-set can be further divided into the two subsets α_{110} and α_{111}.

C_1	0 0·	α_0	0· 0				
C_2	0 1·		0· 1				
C_3	1 0 0·	α_1	$1\cdot \underline{0\ 0}$	α_{10}	1 0· 0		
C_4	1 0 1·		$1\cdot \underline{0\ 1}$		1 0· 1		
C_5	1 1 0 0·		$1\cdot \underline{1\ 0\ 0}$	α_{11}	$1\ 1\cdot \underline{0\ 0}$	α_{110}	1 1 0· 0
C_6	1 1 0 1·		$1\cdot \underline{1\ 0\ 1}$		$1\ 1\cdot \underline{0\ 1}$		1 1 0· 1
C_7	1 1 1 0·		1· 1 1 0		1 1· 1 0	α_{111}	1 1 1· 0
C_8	1 1 1 1·		1· 1 1 1		1 1· 1 1		1 1 1· 1

FIGURE 16. HIERARCHY OF PREFIX SETS IN AN EXHAUSTIVE PREFIX ENCODING

A proper prefix of a code whose terminator β is also a code is called a C-prefix. Thus 1 is a C-prefix of the codes 100, 101, 1100, and 1101 in Figure 16, for the β's are 00, 01, 100, and 101, respectively, each of which is a code in the encoding. Terminators that are codes are underlined in Figure 16.

A code contains a compound C-prefix if it is a member of prefix set α_i and a partition of its terminator forms one or more C-prefixes that are members of α_i. Thus the code 1100 has a compound C-prefix for it can be partitioned into 1· 100 and 100 can be partitioned into 1· 00. Both terminators 100 and 00 are codes and the C-prefix after both partitions is 1.

Terminators can also be classified. A terminator whose digits comprise a code is called a C-terminator. It follows that the terminator of every C-prefix is a C-terminator. A compound C-terminator is a terminator which is associated with more than one prefix set. Thus 00 is a compound C-terminator for it is a terminator of the prefix set α_1 (1· 00) and also of the prefix set α_{11} (11·00). A multiple C-terminator is a terminator that can be partitioned into more than one C-terminator. If 000, 001, and 11000001 are codes of an encoding, 000001 is a multiple C-terminator for, after the partition 11·000001, the terminator can be further partitioned into the two codes 000 and 001.

C. Self-Synchronization

Gilbert and Moore [11] have investigated the self-synchronizing properties of exhaustive prefix encodings. They have found that if an exhaustive prefix encoding has some codes of length greater than 1, if at least two of the codes have the same terminator, and if the greatest common divisor of the lengths of the codes is equal to 1, it is possible for a string of digits to return to synchronism after the loss of the initial digits.

Self-synchronism is illustrated below using the encoding of Figure 16. Original codes are superlined and transformed codes are underlined. It is assumed that the initial binary digit is dropped due to loss of synchronism.

$$\overset{C_1}{\overline{\cancel{0}\,0}}\;\overset{C_7}{\overline{1\,1\,1\,0}}\;\overset{C_3}{\overline{1\,0\,0}}\;\overset{C_2}{\overline{0\,1}}$$
$$\cancel{0}\;\underset{C_2}{\underline{0\,1}}\;\underset{C_6}{\underline{1\,1\,0\,1}}\;\underset{C_1}{\underline{0\,0}}\;\underline{0\,1}$$

It is seen that after three received codes, the string of digits is restored to its original enciphering.

The sequence of codes $C_1 \cdot C_7 \cdot C_3$ will be called a synchronizing sequence. It is that sequence of input codes which is transformed under loss of synchronism before the input enciphering is restored.

The conditions for self-synchronization may now be stated more precisely employing the terminology of the previous section. A sequence of digits is self-synchronizing if at a given location in the sequence one of the following conditions is satisfied:

1. A C-terminator of an input code is an improper prefix of a transformed code.

$$\ldots \underbrace{\overbrace{0\,1}^{\text{C-Prefix}}}_{\text{Term.}} \cdot \underbrace{\overbrace{0\,0\,0\,0}^{\text{C-Term.}}}_{C_1}$$

2. A terminator of an input code is a C-prefix of a transformed code.

$$\ldots \underbrace{\overbrace{0\,1}^{\text{Term.}}}_{\substack{\text{C-}\\\text{Prefix}}} \cdot \underbrace{\overbrace{0\,0\,0\,0}^{C_i}}_{\substack{\text{C-}\\\text{Term.}}}$$

3. A multiple C-terminator of an input code forms two or more improper prefixes of transformed codes.

$$\ldots \underbrace{0\,1\,0\,0\,1}_{\text{Prefix}} \cdot \underbrace{\overbrace{1}^{\substack{\text{C-}\\\text{Prefix}}}}_{\text{Term.}} \cdot \overbrace{\underbrace{0\,0\,1\,0}_{C_{i_m}} \cdot\cdot \underbrace{0\,0\,0\,0}_{C_{i_{m+1}}}}^{\substack{\text{Multiple}\\\text{C-Term.}}}$$

4. A terminator of an input code is a C-prefix of a multiple C-terminator of a transformed code.

$$\ldots \overbrace{0\,1\,0\,0\,1}^{\text{Prefix}} \cdot \underbrace{\overbrace{1}^{\text{Term.}}}_{\substack{\text{C-}\\\text{Prefix}}} \cdot \underbrace{\overbrace{0\,0\,1\,0}^{C_i} \cdot \overbrace{0\,0\,0\,0}^{C_{i_{m+1}}}}_{\substack{\text{Multiple}\\\text{C-Term.}}}$$

Analysis shows that condition 2 is a conjugate of condition 1 in which the partitions of the input and transformed codes are interchanged. Similarly, condition 4 is a conjugate of condition 3. It is also noted that conditions 3 and 4 are similar to conditions 1 and 2 except for the presence of multiple C-terminators in the former conditions.

D. Error Detection by Self-Synchronization

1. Determination of the synchronizing sequence

Let a message of m symbols be of the form

$$M = C_{i_1} \cdot C_{i_2} \cdot \ldots \cdot C_{i_p} \cdot C_{i_q} \cdot \ldots \cdot C_{i_m}$$

where i takes on any value between 1 and n and n is the number of codes in the encoding. If the initial digits of the message are dropped because of loss of synchronism the input codes in the synchronizing sequence will be transformed into legitimate but erroneous codes. Let this transformed message be M'.

$$M' = C_{j_1} \cdot C_{j_2} \cdot \ldots \cdot C_{j_p} \cdot C_{i_q} \cdot \ldots \cdot C_{i_m},$$

where i and j take on any value between 1 and n, and $C_{j_1}, C_{j_2}, \ldots, C_{j_p}$ are transformed codes of Γ arising from the loss of synchronism. At code C_{i_q} the sequence will return to synchronism and the remainder of the message will appear in its original form. The synchronizing sequence, $C_{i_1} \cdot C_{i_2} \cdot \ldots C_{i_p}$, is designated by the symbol σ.

Letting θ designate the code sequence of the original message, when the message is in synchronism, $M = \theta$. When the message is not in synchronism, $M' = \sigma \cdot \theta_r$, where θ_r is the unchanged remainder of the original message. θ_r may similarly consist of a string of digits which contains another synchronizing sequence and a diminished remainder. In general, a message may be represented as

$$M' = \sigma_1 \cdot \sigma_2 \cdot \sigma_3 \cdot \ldots \cdot \theta_r$$

where θ_r may or may not be a synchronizing sequence.

The relationship above states the following: if the initial digits of a message are dropped because of loss of synchronism, σ_1 is a transformed code sequence which returns to the original enciphering at σ_2. If digits are dropped at σ_2, the original message is restored at σ_3, etc.

If the length of the synchronizing sequence σ is known, error detection can be provided for this sequence. By determining the set of all synchronizing sequences in a message an error detection system can be devised which will afford protection for the entire message. It will be demonstrated that such an error detection system is capable not only of detecting loss of synchronism arising from deletion errors, but also of detecting transposition errors (0 to 1 transformations and conversely) and insertion errors arising from spurious signals.

Since each synchronizing sequence is dependent upon a unique concatenation of codes, it is necessary to determine the precise location in a sequence of codes at which self-synchronization occurs. This can be accomplished by considering the enciphering device as a sequential machine. Both Neumann [17] and Gilbert [20] have utilized the theory of sequential machines in their studies of the n-definite and prefix synchronized encodings.

The enciphering device, designated machine T, is a finite-state sequential machine consisting of a finite number n of states, a finite number m of possible input symbols, and a finite number p of possible output symbols.

The m input symbols are the codes of Γ. Two output symbols $Z = \left\{ S, \overline{S} \right\}$ are employed: S which denotes return to synchronism and $\overline{S}$, the complement of S, which denotes non-return to synchronism. The states comprise the set $Q = \left\{ q_1, q_2, \ldots, q_n \right\}$.

The states of the machine fall into three classes:

1. S state (synchronism). An input code is left unchanged.
2. xF state (subtract x digits). A fixed number of digits is subtracted from an input code. The number of digits subtracted is designated by an integer x.
3. D_i state (dependent). The number of digits that are subtracted from an input code varies in accordance with the prefix of the input code. There are i states of D as determined by a particular encoding.

An output function Λ associates with each state q_i and each input C_j an output z_k, i. e., $z_k = q_i \Lambda C_j$. A next state function Δ associates with each (q_i, C_j) - pair a state $q_r = q_i \Delta C_j$. Thus the state the machine will be in at a given time is dependent only on its state at the previous time and the previous input. The output symbol at a given time is dependent only upon the current state of the machine.

The sequential machine is completely specified, i. e., the next state function and the output function are defined for all input symbols and all states. However, the machine is loosely connected since it is not possible to reach every state from every other state. It is the latter property of machine T which makes possible the determination of a synchronizing sequence.

Machine T is completely defined by a flow table or a state diagram. The state diagram is a directed graph, the nodes of which correspond to the states and the edges correspond to the possible transitions between states. An arrowhead on a branch designates the direction of the transition. Input codes which bring about the transition are designated beside each edge. Each encoding will be described by a unique sequential machine.

A flow table and its associated state diagram describing the encoding of Figure 16 are shown in Figure 17 . The table was derived by analyzing the encoding and defining its possible states.

The next state for each element of the flow table is determined by the next state function. A state may be considered as an operator that effects a transformation on an input code. It is obvious that if the machine is in state S an input code will be left intact and the next state will also be S.

In the F state, one binary digit is subtracted from an input code. If the input sequence of binary digits is read from left to right this is equivalent to deleting the first left-hand binary digit of an input code. The next state can be determined by performing this operation on each input code and juxtaposing with each of the resulting transformed codes all codes of the encoding. The sequences for each code are then deciphered.

The next state $q_i = F \triangle C_1$, for example, is determined as follows:

$$\emptyset\ \underline{0\cdot 0}\ 0$$
$$\emptyset\ \underline{0\cdot 0}\ 1$$
$$\emptyset\ \underline{0\cdot 1}\ \underline{0\ 0}$$
$$\emptyset\ \underline{0\cdot 1}\ \underline{0\ 1}$$
$$\emptyset\ \underline{0\cdot 1}\ \underline{1\ 0\ 0}$$
$$\emptyset\ \underline{0\cdot 1}\ \underline{1\ 0\ 1}$$
$$\emptyset\ \underline{0\cdot 1}\ 1\ 1\ 0$$
$$\emptyset\ \underline{0\cdot 1}\ 1\ 1\ 1$$

It is seen that the next state is F, for all transformed codes derived by juxtaposing every code of the encoding with 0 are formed by subtracting one

NEXT STATE								
PRESENT STATE	INPUT							
	C_1 00	C_2 01	C_3 100	C_4 101	C_5 1100	C_6 1101	C_7 1110	C_8 1111
S	S	S	S	S	S	S	S	S
F	F	D	S	S	S	S	F	F
D	S	S	S	S	F	D	F	D

OUTPUT	
NEXT STATE	OUTPUT
S	S
F	$\bar{S}$
D	$\bar{\bar{S}}$

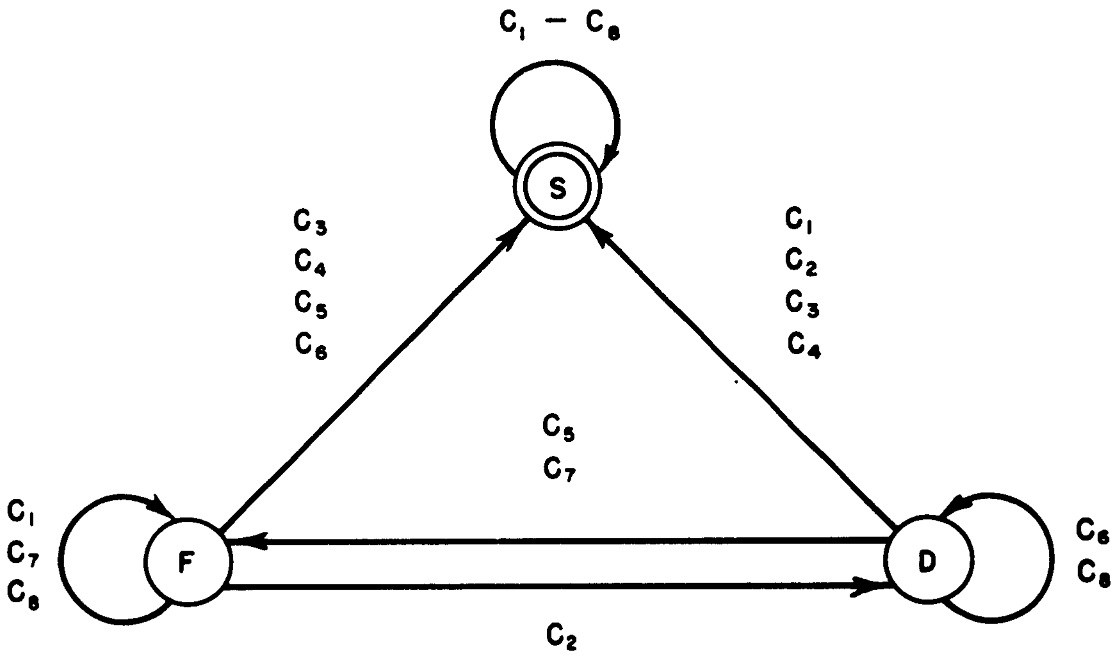


FIG. 17 SELF-SYNCHRONIZATION STATE DIAGRAM FOR 8-CODE EXHAUSTIVE PREFIX ENCODING

binary digit from the input code.

The next state $q_{i+1} = F \triangle C_2$ is determined in a similar manner:

$\emptyset\ \underline{1\cdot 0\ 0}$

$\emptyset\ \underline{1\cdot 0\ 1}$

$\emptyset\ \underline{1\cdot 1\ 0\ 0}$

$\emptyset\ \underline{1\cdot 1\ 0\ 1}$

$\emptyset\ \underline{1\cdot 1\ 1\ 0}\ 0$

$\emptyset\ \underline{1\cdot 1\ 1\ 0}\ 1$

$\emptyset\ \underline{1\cdot 1\ 1\ 1}\ 0$

$\emptyset\ \underline{1\cdot 1\ 1\ 1}\ 1$

It is now noted, however, that either two or three binary digits are subtracted from an input code depending on whether the prefix of the input code is 0 or 1. Hence, the next state is D.

The next state entries in the table for the remaining input codes when the machine is in state F, can be determined as above.

The next state $q_r = D \triangle C_j$ is determined by juxtaposing each input code in turn to the binary digit 1:

$\underline{1\cdot 0\ 0}$

$\underline{1\cdot 0\ 1}$

$\underline{1\cdot 1\ 0\ 0}$

$\underline{1\cdot 1\ 0\ 1}$

$\underline{1\cdot 1\ 1\ 0}\ 0$

$\underline{1\cdot 1\ 1\ 0}\ 1$

$\underline{1\cdot 1\ 1\ 1}\ 0$

$\underline{1\cdot 1\ 1\ 1}\ 1$

In accordance with the definition of the D state, two or three binary digits are subtracted from the input code to form a transformed code. The next state for codes C_1 through C_4 is the S state. The remaining binary digits, after forming codes from C_5 through C_8, are either 0 or 1. Since it was previously determined that a remainder of 0 places the machine in state F and a remainder of 1 places the machine in state D, the next state is known for the remaining codes.

We now define a state q_o as the initial state of the machine such that if there have been no inputs, the machine is in state q_o. The synchronizing sequence of a message is obtained by starting the machine in state q_o and processing a sequence of input codes in order of their enciphering until the output S is obtained. The convention will be adopted that q_o = F (any state other than S could be selected).

From Figure 17 the synchronizing sequence for the following message is determined:

$$\overset{C_7}{\overline{1\;1\;1\;0}}\;\overset{C_2}{\overline{0\;1}}\;\overset{C_6}{\overline{1\;1\;0\;1}}\;\overset{C_5}{\overline{1\;1\;0\;0}}\;\overset{C_1}{\overline{0\;0}}\;\overset{C_7}{\overline{1\;1\;1\;0}}\;\overset{C_3}{\overline{1\;0\;0}}\;\overset{C_4}{\overline{1\;0\;1}}$$

Starting in state F, and following the transitions of the flow table, the following output table is obtained:

Present State	F	F	D	D	F	F	F
Input	C_7	C_2	C_6	C_5	C_1	C_7	C_3
Next State	F	D	D	F	F	F	S
Output	$\bar{S}$	$\bar{S}$	$\bar{S}$	$\bar{S}$	$\bar{S}$	$\bar{S}$	S

That the message does return to synchronism at C_4 after losing the initial digit is verified below, condition 1 for self-synchronization being satisfied.

$$\begin{array}{cccccccccccccccccccccccccc}
\multicolumn{4}{c}{C_7} & \multicolumn{2}{c}{C_2} & \multicolumn{4}{c}{C_6} & \multicolumn{4}{c}{C_5} & \multicolumn{2}{c}{C_1} & \multicolumn{4}{c}{C_7} & \multicolumn{3}{c}{C_3} & \multicolumn{3}{c}{C_4} \\
\cancel{X} & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\
 & \multicolumn{4}{c}{C_5} & \multicolumn{4}{c}{C_7} & \multicolumn{4}{c}{C_7} & \multicolumn{2}{c}{C_1} & \multicolumn{2}{c}{C_2} & \multicolumn{4}{c}{C_6} & \multicolumn{2}{c}{C_1} & \multicolumn{3}{c}{}
\end{array}$$

As the number of codes in an encoding is increased the flow table and its associated state diagram become more complex. The self-synchronization flow table and state diagram for a 9-code encoding are shown in Figure 18. Although only one code has been added to the encoding, five states are now required to describe the sequential machine. In the D_1 state, either one or two digits are partitioned from the input code; in the D_2 state, either two or three digits are partitioned.

It is estimated that a 5,208-code encoding requires 16 F states and 23 D states in addition to the S state to describe its self-synchronization flow table. It is obvious that a 40 x 5,208 matrix is well on the way to posing formidable problems in the way of handling. By employing the analysis of code partitions and the structure functions of the encoding, however, it is possible to determine the synchronizing sequence in a simple and straight-forward manner without recourse to a flow table.

Let us first examine the structure functions of the non-spaced exhaustive prefix encoding. Huffman [10] has described a procedure for determining the lengths of codes in an encoding and Gilbert and Moore [11] have shown that the Huffman encoding has a cost which is less then or equal to that of any uniquely decipherable encoding for a given alphabet (Theorem 11). A computer program for generating a frequency tree and determining the structure functions (the number of codes of length L called the B-set) is described in Section IV-B.

PRESENT STATE	NEXT STATE								
	INPUT								
	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9
	00	010	011	100	101	1100	1101	1110	1111
S	S	S	S	S	S	S	S	S	S
F	D_1	F	2F	S	F	S	S	F	F
2F	S	D_1	D_2	D_1	D_2	S	F	F	2F
D_1	D_1	F	2F	D_1	D_2	S	F	F	2F
D_2	S	D_1	D_2	S	S	D_1	D_2	D_1	D_2

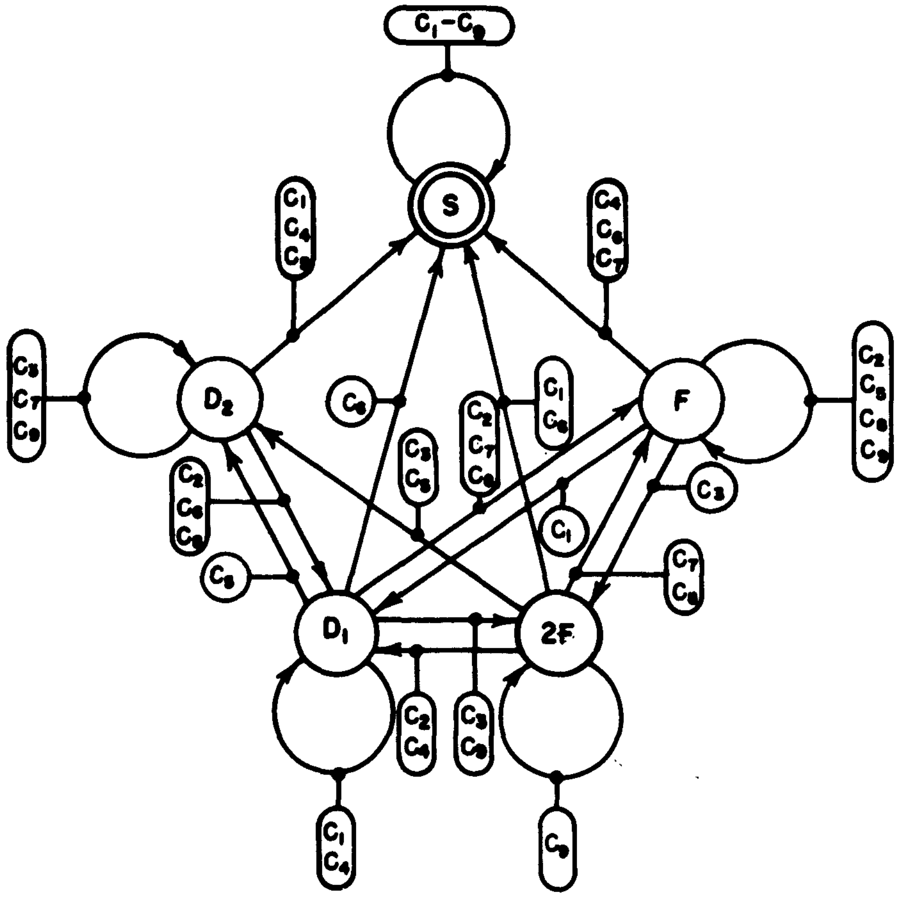


FIG. 18. SELF-SYNCHRONIZATION STATE DIAGRAM FOR 9-CODE EXHAUSTIVE PREFIX ENCODING

Given an optimum structure function, a canonical encoding can be constructed in which the numerical values of the codes are monotone increasing and each code has the smallest possible numerical value consistent with the requirement that the code be an improper prefix. A computer routine generates codes by assigning a sequence of zeros to the first non-empty B-set, the number of zeros corresponding to the minimum length code. Subsequent codes are assigned in accordance with the following rules:

1. Add binary one to the preceding code until the next B-set is reached;
2. Add binary 10 for the first code of a new B-set and return to step 1;
3. If a B-set is null, add 0 to the preceding code and return to step 1.

The set of generated codes is dense and storage efficiency is 100 percent since it is possible to use every storage location in a block in the decoding and deciphering process. Decoding entails the decomposition of a sequence of binary digits into its constituent codes. Deciphering associates each code with the symbol it represents.

The decoding procedure utilizes the upper limits of each B-set in a breakpoint table and by examining the incoming digits one by one and comparing the sequence that is formed with the breakpoint table, the number of digits in a received code is determined. By associating a base address with the initial code of each B-set, the address of each code is readily computed for deciphering.

The synchronizing sequence can be determined by deciphering the encoded message at the transmitter with the machine in state $q_o = F$. In a computer, this process is accomplished by processing the input codes, in the order in which they appear in the message, through two registers, designated A and B. The first code is placed in register A and the initial digit is deleted to place the transmitter in state F (minus 1 digit). The remaining digits are shifted one at a time starting at the left into the right hand position of register B. After each shift, it is determined whether or not register B contains a code. The same computer routine that is used to decipher the string of binary digits at the receiver is employed to test the contents of the B register. When register A is emptied, the next code is placed in it. When a code is formed in register B, 'he register is cleared and the process is continued until both A and B registers are simultaneously empty, indicating that one of the conditions for self-synchronization have been satisfied and that the machine is in state S.

The process is illustrated in Figure 19 for the synchronizing sequence $C_1 \cdot C_7 \cdot C_3$. Shifted digits in the A register are marked by slashes. Codes formed in the B register are transformed codes and are underlined while input codes are superlined.

	B REGISTER	A REGISTER	
		$\overline{\cancel{0}\,0}$	(C_1)
	0	$\cancel{0}$	
(C_2)	$\underline{0\ 1}$	$\overline{\cancel{1}\,1\,1\,0}$	(C_7)
	1	$\cancel{1}\ 1\ 0$	
	1 1	$\cancel{1}\,0$	
	1 1 0	$\cancel{0}$	
(C_6)	$\underline{1\ 1\ 0\ 1}$	$\overline{\cancel{1}\,0\,0}$	(C_3)
	0	$\cancel{0}\,0$	
(C_1)	$\underline{0\ 0}$	$\cancel{0}$	
	- (SYNCHRONISM)	-	

FIGURE 19. DETERMINATION OF SYNCHRONIZING SEQUENCE WITH A AND B REGISTERS.

2. Error Detection

a. General Theory

Let us now consider a second machine, called machine R, which is a receiving device. Machines R and T will be isomorphic if they have the same behavior and are, therefore, indistinguishable. The machines will be indistinguishable if for every state of T and every sequence of inputs, there exists a state of R such that the sequence of inputs beginning with machine T in state q_i produces an output identical to that produced by machine R beginning in state q_j with the same sequence of inputs, and conversely [21].

In mathematical terms, using unprimed symbols to refer to machine T and primed symbols to refer to machine R, $T \cong R$ if $\Gamma = \Gamma'$, $Z = Z'$ and there is a one-to-one correspondence $q \longleftrightarrow q'$ between Q and Q' such that $q \longleftrightarrow q'$ and $C \longleftrightarrow C'$ implies $q \bigtriangleup C \longleftrightarrow q' \bigtriangleup C'$. Similarly, $q \wedge C \longleftrightarrow q' \wedge C'$.

Machine R will determine the synchronizing sequence of the received binary digits by a process of double deciphering. After a code is obtained from the received sequence of binary digits it is placed in the A register. The initial binary digit is deleted to set the machine in state q_o and the code is placed in the B register. Successive codes are placed simultaneously in both A and B registers and deciphering continues in both registers until both are simultaneously empty.

If machine T starts in state q_o and processes a particular sequence of codes, its output will be S at a distinct location in the sequence. If machine R, which is isomorphic to machine T, begins in the same state and processes the same sequence of codes, its output will be S at the same location in the sequence as with machine T. If the output at the same location in the sequence differs between the machines or the state S is reached at different locations in the sequence by the two machines, it is an indication that an error has occurred in the input codes.

To detect errors it is necessary to mark the location at which the output S is obtained. This is accomplished by designating a particular code of the encoding as a marker and establishing the rule that this marker can only follow a code for which the next state is S. Machine T, for every given sequence of codes, will determine the synchronizing sequences and will insert the marker code following every state S.

The output of a synchronizing sequence of machine T will be designated S_o. Every S_o in machine T will be followed by a marker code. If machine R decodes a marker code when the machine is not in state S, an error is indicated. If machine R arrives at state S and a marker code does not follow, an error is indicated. The latter output will be designated S_ε

since it is a synchronizing sequence for a transformed or erroneous message. In both of the above cases machines T and R are distinguishable.

Other cases of possible error arise from the phenomenon of pseudo-isomorphism. Pseudo-isomorphism results from the property of a sequence of binary digits to self-synchronize at the same location as does the correct sequence for different error patterns. This property, in turn, arises from the many possible ways a string of binary digits can be partitioned.

The pattern of self-synchronization of message sequences is shown in Figure 20. Message A is carried through 25 runs successively dropping the initial digit in each run. Underlined portions indicate that part of a deciphered message which is in synchronism with the immediately preceding message. The example may also be interpreted in a different manner, namely, the message that results after dropping the initial number of binary digits indicated by the run number. Thus, WAS FOOD etc. (run 2) is the message produced by dropping one binary digit from the message EVEN AND. etc. (run 1) or two binary digits from the original message.

(The circumflex denotes synthesis of a complex word by joining a suffix to a free form. A / denotes an elided final E and a macron over a Y indicates it is changed to I.)

It is noted that runs 1, 3, 4, 5, 8-14, 16, 19, 20, 21, 24, and 25 all return to synchronism at the same location in the message. It is also interesting to note that there is an upper bound for self-synchronization of the message shown with from 1 through 25 initial digits dropped, all sequences returning to synchronization by the word NUMBER. The appearance of recurring sub-sequences may also be observed as LIKE WITH in runs 1, 2, 8, 11, 16, and 20, WHAT DO BY in runs 4, 10, and 14, and

MESSAGE A

INCREASE D EFFICIENCY OF AN INFORMATION SYSTEM IS ACHIEVE D BY REDUC~~E~~ ING THE NUMBER

1. EVEN AND. LIKE WITH. OF AN INFORMATION SYSTEM IS ACHIEVE D BY REDUC~~E~~ ING THE NUMBER

2. WAS FOOD LIKE WITH. OF AN INFORMATION SYSTEM IS ACHIEVE D BY REDUC~~E~~ ING THE NUMBER

3. MADE D EFFICIENCY OF AN INFORMATION SYSTEM IS ACHIEVE D BY REDUC~~E~~ ING THE NUMBER

4. TO. WHAT DO BY OF AN INFORMATION SYSTEM IS ACHIEVE D BY REDUC~~E~~ ING THE NUMBER

5. HIM D EFFICIENCY OF AN INFORMATION SYSTEM IS ACHIEVE D BY REDUC~~E~~ ING THE NUMBER

6. OURS. EFFICIENCY OF AN INFORMATION SYSTEM IS ACHIEVE D BY REDUC~~E~~ ING THE NUMBER

7. PRODUCTION. EFFICIENCY OF AN INFORMATION SYSTEM IS ACHIEVE D BY REDUC~~E~~ ING THE NUMBER

8. FOOD LIKE WITH. OF AN INFORMATION SYSTEM IS ACHIEVE D BY REDUC~~E~~ ING THE NUMBER

9. ED WITH EFFICIENCY OF AN INFORMATION SYSTEM IS ACHIEVE D BY REDUC~~E~~ ING THE NUMBER

10. . WHAT DO BY OF AN INFORMATION SYSTEM IS ACHIEVE D BY REDUC~~E~~ ING THE NUMBER

11. AND. LIKE WITH. OF AN INFORMATION SYSTEM IS ACHIEVE D BY REDUC~~E~~ ING THE NUMBER

12. SHE TROUBLE BY OF AN INFORMATION SYSTEM IS ACHIEVE D BY REDUC~~E~~ ING THE NUMBER

13. D EFFICIENCY OF AN INFORMATION SYSTEM IS ACHIEVE D BY REDUC~~E~~ ING THE NUMBER

14. WHAT DO BY OF AN INFORMATION SYSTEM IS ACHIEVE D BY REDUC~~E~~ ING THE NUMBER

15. , TROUBLE BY OF AN INFORMATION SYSTEM IS ACHIEVE D BY REDUC~~E~~ ING THE NUMBER

16. . LIKE WITH. OF AN INFORMATION SYSTEM IS ACHIEVE D BY REDUC~~E~~ ING THE NUMBER

17. AND TO TALK HEAD THE HIM SHANTY ON ME AS GREAT SOMETIMES TO IT A AND IN NUMBER

18. WITH EFFICIENCY OF AN INFORMATION SYSTEM IS ACHIEVE D BY REDUC~~E~~ ING THE NUMBER

19. TROUBLE BY OF AN INFORMATION SYSTEM IS ACHIEVE D BY REDUC~~E~~ ING THE NUMBER

20. LIKE WITH. OF AN INFORMATION SYSTEM IS ACHIEVE D BY REDUC~~E~~ ING THE NUMBER

21. . EFFICIENCY OF AN INFORMATION SYSTEM IS ACHIEVE D BY REDUC~~E~~ ING THE NUMBER

22. TO TALK HEAD THE HIM SHANTY ON ME AS GREAT SOMETIMES TO IT A AND IN NUMBER

23. DO BY OF AN INFORMATION SYSTEM IS ACHIEVE D BY REDUC~~E~~ ING THE NUMBER

24. SNARL OF AN INFORMATION SYSTEM IS ACHIEVE D BY REDUC~~E~~ ING THE NUMBER

25. EFFICIENCY OF AN INFORMATION SYSTEM IS ACHIEVE D BY REDUC~~E~~ ING THE NUMBER

FIGURE 20. SELF-SYNCHRONIZATION OF MESSAGE SEQUENCES WITH SUCCESSIVE DELETION OF INITIAL BINARY DIGITS.

MESSAGE B

THE OBJECTIVE OF THIS RESEARCH PROGRAM

1. HAD PLAYER, THIS RESEARCH PROGRAM
2. DRESS HUNDRED THIS RESEARCH PROGRAM
3. WILL YOU THAT OF BEEN OLD PROGRAM
4. OBJECTIVE OF THIS RESEARCH PROGRAM
5. FISHERMAN THIS I FOOLISH PROGRAM

MESSAGE C

THIS IS A TEST DESIGN ED TO DETERMINE THE PROPERTY ES OF VARY ABLE LENGTH CODE S..

1. , SAID WAS NEXT THE AT AND I IF TYPEWRITER THE PROPERTY ES OF VARY ABLE LENGTH CODE S .
2. . . , EVER NOT IN AT AND I IF TYPEWRITER THE PROPERTY ES OF VARY ABLE LENGTH CODE S .
3. THE. AS ISH WAS AS FILL. GALLANT. THEY CASE AND % BY MONEY WAS YOUR. HUNT S .
4. OF EVER NOT IN AT AND I IF TYPEWRITER THE PROPERTY ES OF VARY ABLE LENGTH CODE S .
5. SAID WAS NEXT THE AT AND I IF TYPEWRITER THE PROPERTY ES OF VARY ABLE LENGTH CODE S.

MESSAGE D

ACHIEVEMENT OF A REDUCTION IN THE NUMBER OF SYMBOL S REQUIRE D TO SEND A MESSAGE

1. SHORE OF A REDUCTION IN THE NUMBER OF SYMBOL S REQUIRE D TO SEND A MESSAGE
2. ART OF A REDUCTION IN THE NUMBER OF SYMBOL S REQUIRE D TO SEND A MESSAGE
3. AMERICAN THE, WOULD PERCENTAGE THE THE NUMBER OF SYMBOL S REQUIRE D TO SEND A MESSAGE
4. NOT, OR WASHINGTON BALL THE NUMBER OF SYMBOL S REQUIRE D TO SEND A MESSAGE
5. MURDER, ENGAGEMENT THOUGH. HIS SHE UP(P) ING OCCUR SEND. COULD CONFUSION A MESSAGE

FIGURE 20. SELF-SYNCHRONIZATION OF MESSAGE SEQUENCES WITH SUCCESSIVE DELETION OF INITIAL BINARY DIGITS (CONTINUED).

the longer transformed sequence which appears in run 17 and again in run 22. The word NUMBER remains the upper bound for self-synchronization of this sequence through run 50.

The appearance of recurring sub-sequences and the existence of a bound for the self-synchronization of messages when the initial digits are successively dropped are also shown in messages B, C, and D, of Figure 20.

Pseudo-isomorphism leads to two cases in which machines T and R are indistinguishable although they are no longer truly isomorphic. In one case, it is possible to obtain a sequence of codes whose output is S at the proper location. The output is then seemingly S_o and is followed by a marker code although the sequence differs from the original sequence due to loss of synchronism or other errors. In the second case, the marker code itself is generated from a combination of errors. If the machine is not in state S when the marker code is deciphered, the error is evident. If the machine is in state S, because it is followed by the marker code, the output is again seemingly S_o, and the error is again hidden.

Pseudo-isomorphism is illustrated in Figure 21 with the seven code synchronizing sequence from the 8-code encoding previously described. T_o is the sequence formed at the transmitter by starting machine T in state F (minus one digit) and indicates the location of S_o. If machine R receives sequence R_1 instead of T_o, it will return to synchronism at the point indicated. The error will be detected since a marker code is not encountered and hence S is designated. However, if R_2 is received at machine R and its synchronizing sequence is determined, it is found that self-synchronization occurs at the required location and S_o is designated, although errors did occur in the sequence. Similarly, if other than the initial binary digits are deleted, S_o can be obtained as shown in R_3 and R_4. A transposition error (encircled) in R_5 also leads to the S_o condition in spite of the error.

It is this property of self-synchronization that makes the choice of an initial state for error detection optional since the initial state is not in any way related to an expected type of error. It is only required that machines T and R both start in the same initial state.

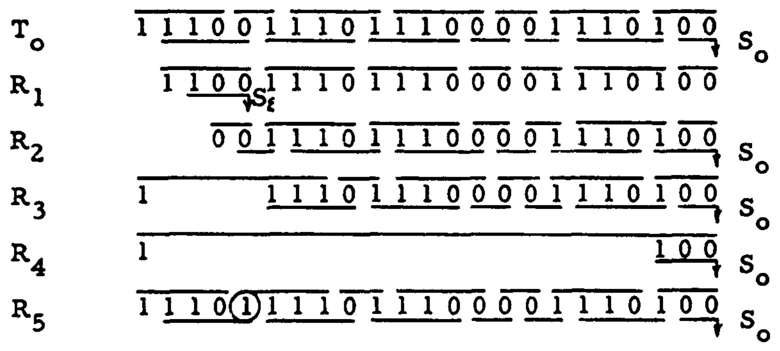

T_o 1 1 1 0 0 1 1 1 0 1 1 1 0 0 0 0 1 1 1 0 1 0 0 S_o
R_1 1 1 0 0 1 1 1 0 1 1 1 0 0 0 0 1 1 1 0 1 0 0 S_ϵ
R_2 0 0 1 1 1 0 1 1 1 0 0 0 0 1 1 1 0 1 0 0 S_o
R_3 1 1 1 1 0 1 1 1 0 0 0 0 1 1 1 0 1 0 0 S_o
R_4 1 1 0 0 S_o
R_5 1 1 1 0 1 1 1 1 0 1 1 1 0 0 0 0 1 1 1 0 1 0 0 S_o

FIGURE 21. PSEUDO-ISOMORPHISM IN SELF-SYNCHRONIZING SEQUENCES.

To detect errors under the condition of pseudo-isomorphism, a sequence of check digits is used in conjunction with the marker code. Since the marker code is unique, it can be used to frame the check digits which need not be part of the exhaustive prefix encoding [17] . For this reason, the marker will be called the sync frame code (SFC) and will be followed by a designated number of check digits called the CHECKWORD.

The check digits can be selected in keeping with the self-synchronizing properties of a particular encoding, the characteristics of a communications channel, and the error probabilities associated with the channel. The number of check digits is dependent upon the reliability to be achieved, namely, the probability of undetected errors arising from pseudo-isomorphism.

The location of the CHECKWORD, preceding the sequence which it checks or following the sequence (but always framed by the SFC), is a matter of significance. Let us first consider placing the CHECKWORD following the sequence that is checked. If an error occurs in a sync frame code or in a preceding sequence in such a manner that the SFC is transformed by an erroneous partition, a CHECKWORD that follows the SFC will be lost since it will no longer be framed. In this case, decoding and the error detection procedure must continue into succeeding sequences until an error is detected. False generation of the SFC followed by a specified number of digits construed as check digits can also give rise to a condition in which an error may be undetected and may not be confined to the synchronizing sequence.

On the other hand, if the CHECKWORD precedes the synchronizing sequence with which it is associated, it is necessary for an error to occur within the few digits of the CHECKWORD to transform it. Parity checks on the CHECKWORD itself can be designed to detect errors within the CHECKWORD.

The sync frame code should be selected so that its length is a function of its estimated frequency of occurrence. Since a falsely generated SFC will usually indicate an error, the particular code selected should be a commonly occurring sequence of digits.

b. The Free-Running Self-Synchronizing System

The free-running self-synchronizing error detection system checks errors in the naturally occurring synchronizing sequences as determined by the sequential machines. The length of the blocks of digits being checked will therefore vary in accordance with the length of the synchronizing sequences.

The error detection system operates in the following manner. At machine T, a message is started with a sync frame code. After each symbol of the input message is enciphered, the digits are processed through the A and B registers and synchronizing sequence σ_1 is determined. The checkword is computed and transmitted followed by σ_1 and a second SFC. σ_2 is then determined and the process is continued until the end of message is reached. The string of binary digits emitted from machine T appears as

SFC	CHECKWORD (σ_1)	σ_1	SFC	CHECKWORD (σ_2)	σ_2	SFC	. . .

At machine R the previously described procedure for determining a synchronizing sequence is followed, starting in the same initial state as machine T and the received bit string is deciphered through the A and B registers. An error signal results if any one of the following conditions is detected:

1. Initial code is not SFC
2. $\bar{S}$ followed by SFC
3. S not followed by SFC (S_ϵ)
4. S followed by SFC (S_o); CHECKWORD fails.

The error signal results in a request for retransmission of the string of digits following the last correctly received sequence terminating in an SFC.

Examples of error detection for the 9-code encoding of Figure 18 are shown in Figure 22. The top line, marked T_o, shows a string of binary digits emitted at the transmitter. The lines designated R_i indicate messages received with various configurations of errors. The type of error is designated by a symbol above the digit (s) which are in error: $\textcircled{\downarrow}$ = deletion, $\dagger$ = insertion, $\downarrow$ = transposition error.

Each sequence is preceded by a sync frame code, designated as 101, followed by four check digits, b_1 to b_4. The CHECKWORD is computed as follows:

$b_1 \cdot b_2$ = number of binary digits in σ_i (mod 4)

b_3 = number of 0's in σ_i (mod 2)

b_4 = number of 1's in σ_i (mod 2)

From the construction of the CHECKWORD, four error checks can be made at machine R.

1. $(b_3 + b_4) \bmod 2 \neq b_2$
2. $b_1 \cdot b_2 \pmod 4 \neq$ number binary digits in σ_{i_R}
3. $b_3 \neq$ number 0's in σ_{i_R}
4. $b_4 \neq$ number 1's in σ_{i_R}.

The first check is a test of the CHECKWORD itself for the sum of the zero and one parity check digits (modulo 2) must equal the number of check digits of the length of the synchronizing sequence (mod 2). Since a CHECKWORD is received before a synchronizing sequence, this check can detect a class of errors before deciphering of the message sequence itself is begun.

The error in R_1 is detected because the initial code is not an SFC. Sequences R_2 - R_5 illustrate single and multiple deletion errors which destroy the original checkword. Sequences R_6 - R_9 illustrate deletion errors restricted to the synchronizing sequence. R_6 and R_7 illustrate the effect of a compound C-prefix, for partitioning the initial code 1100 into 1· 100 and 11· 00 causes an error fault at the same location regardless of whether one or two digits are deleted. R_8 illustrates the generation of a false SFC.

R_{10} - R_{17} are sequences with single and double transposition errors. R_{10} shows a single transposition error in the checkword, R_{11} shows the error in the terminal SFC, and R_{12} - R_{15} show single transposition errors in the synchronizing sequence. The effects of double transposition errors in σ are indicated in R_{16} and R_{17}

An insertion error is shown in R_{18}. Several configurations of compound errors are illustrated in R_{19} - R_{22}.

In the event that the error configuration results in transformation of the SFC which terminates a synchronizing sequence and synchronism is not restored at or before this point in the sequence, the deciphering routine will proceed through the transformed SFC, the following CHECKWORD, and the next synchronizing sequence. This situation is illustrated in Figure 23. T_o designates the sequence of binary digits emitted at machine T for two

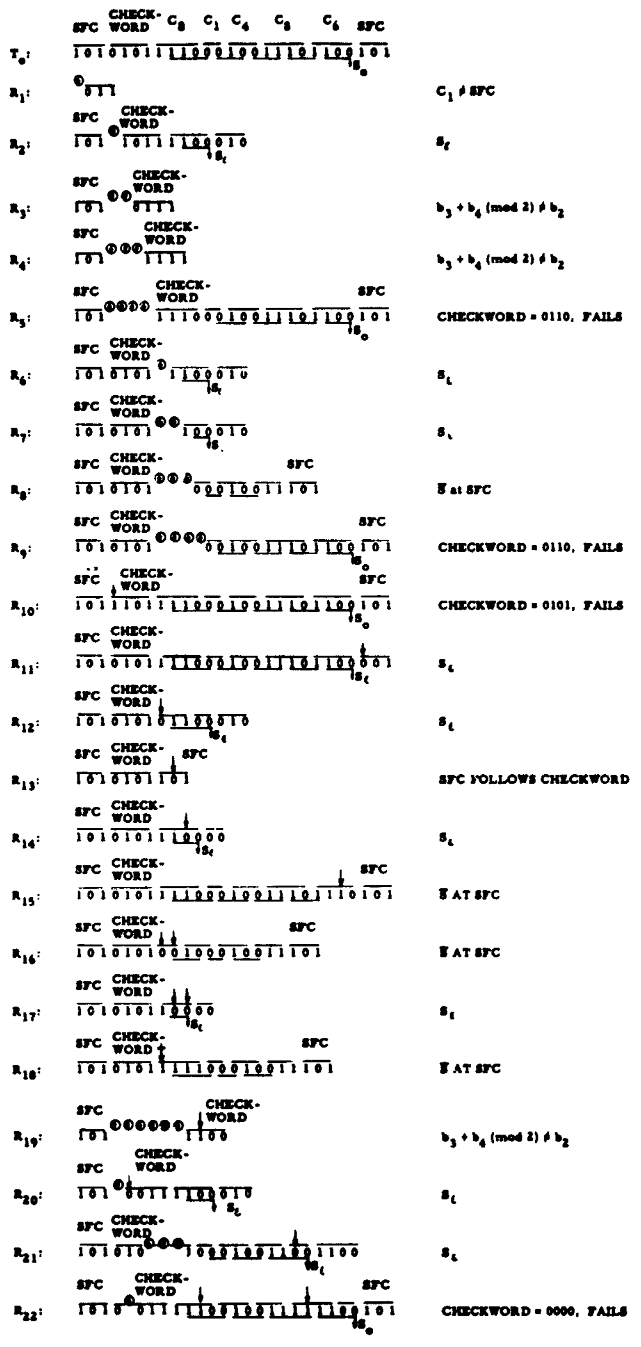


FIGURE 22. DETECTION OF SINGLE, MULTIPLE, AND COMBINED ERROR CONFIGURATIONS IN FREE-RUNNING SYSTEM.

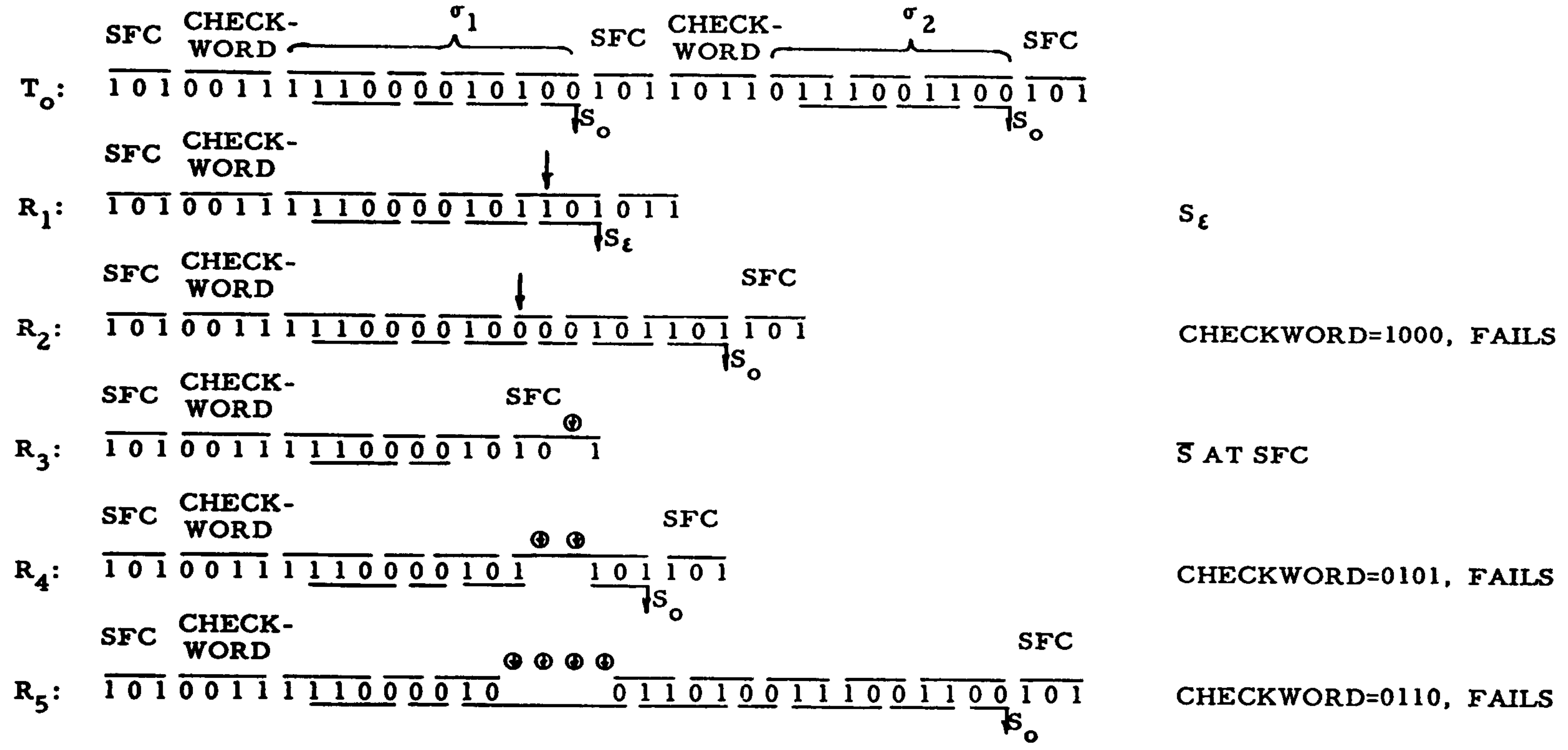

SFC CHECK-WORD σ1 SFC CHECK-WORD σ2 SFC
T0: 101 00111 1110000010100 101 10110 1110011100 101
S0
SFC CHECK-WORD
R1: 101 00111 11100000101101011
Sε
SFC CHECK-WORD SFC
R2: 101 00111 11100000100000101101 101
S0
CHECKWORD=1000, FAILS
SFC CHECK-WORD SFC
R3: 101 00111 1110000010 10 1
S̄ AT SFC
SFC CHECK-WORD SFC
R4: 101 00111 1110000101 101 101
CHECKWORD=0101, FAILS
SFC CHECK-WORD SFC
R5: 101 00111 111000010 0110100111001100 101
CHECKWORD=0110, FAILS

FIGURE 23. ERROR DETECTION WHEN ERRORS TRANSFORM TERMINAL SFC OF SYNCHRONIZING SEQUENCE.

synchronizing sequences. At R_1 the SFC following σ_1 is lost due to a transposition error. The error is detected, however, since the state S_ϵ is reached. In R_2 the transmitted SFC following σ_1 is again lost but failure of the CHECKWORD upon generation of a false SFC detects the error. R_3 and R_4 illustrate deletion errors. The sequence in R_5 synchronizes at the proper location following σ_2 although the SFC after σ_1 was lost; failure of the CHECKWORD indicates this error. Since retransmission will be requested from the last acceptable SFC, the entire sequence of σ_1 and σ_2 will be retransmitted.

In a finite sequence of binary digits it is possible that, θ_r, the remainder of the original message after the synchronizing sequences have been formed does not form a synchronizing sequence. Since the message does not self-synchronize at the final digit it is necessary to force synchronization by affixing a finite number of digits to the sequence.

Forced synchronization is first discussed in a general sense. Consider a sequential machine T whose internal state is determined at the n-th binary digit of an incoming string of codes. The state of the machine can be determined at b_n, the n-th binary digit, by examining the contents of the A and B registers. The possible configurations at b_n can be enumerated in terms of these contents. An empty A register will be designated A0 and an A register containing binary digits will be designated A1. Simularly, an empty B register will be B0 and a B register containing binary digits will be B1. It is obvious that four configurations can exist at b_n, only one of which indicates that the machine is in state S. These are shown below with examples from the 8-code encoding.

CONFIGURATION	STATE	EXAMPLE
		$b_1\ b_2\ \ \ldots\ \ b_n\ \ \ldots$
A0-B0	S	0 1 1 0 0
A0-B1	$\overline{S}$	1 1 1 0 0
A1-B0	$\overline{S}$	0 1 1 1 0 | 1
A1-B1	$\overline{S}$	1 1 1 | 0 0

The A0-B0 configuration is the synchronized state. Code sequences in the $\overline{S}$ configurations at b_n can be analyzed by the method of partitions. For configuration A0-B1, the contents of the B register can consist of any terminating string which is not a C-terminator. From Figure 17 , it is seen that 0, 1, 10, 11, 110, and 111 are not C-terminators. It is also seen that the 0 and 1 terminating string can be produced by four partitions. Thus 0 is the terminating string of the following codes: 0· 0, 10· 0, 110· 0, and 111· 0, while 1 is the terminating string of 0· 1, 10· 1, 110· 1, and 111· 1. The other four terminating strings each arise from one partition. There are a total of 12 A0-B1 sequences as shown in Figure 24.

The digits added to the A0-B1 configuration must comprise a code because synchronization occurs only when the final digits of an input code and a transformed code are similar and since the A register is empty at b_n, only a code can be placed in the register. The digits remaining in the B register can be considered as a prefix of the code to be added. If it is a C-prefix, addition of a C-terminator will produce synchronization. If it is not a C-prefix, a code must be added which can be partitioned in such a manner that a C-prefix is formed which will at the same time be the terminator of the prefix in the B register.

Consider the first A0-B1 sequence of Figure 24, where a 0 remains in the B register. A perusal of Figure 17 reveals that there is no C-prefix in the α_0 prefix set and that the two codes in the α_0 set have terminators 0 and 1. The 0 terminator can now be ruled out and it is seen that 1 is a C-prefix of the code 100. Hence, affixing 100 to the sequence will result in self-synchronization.

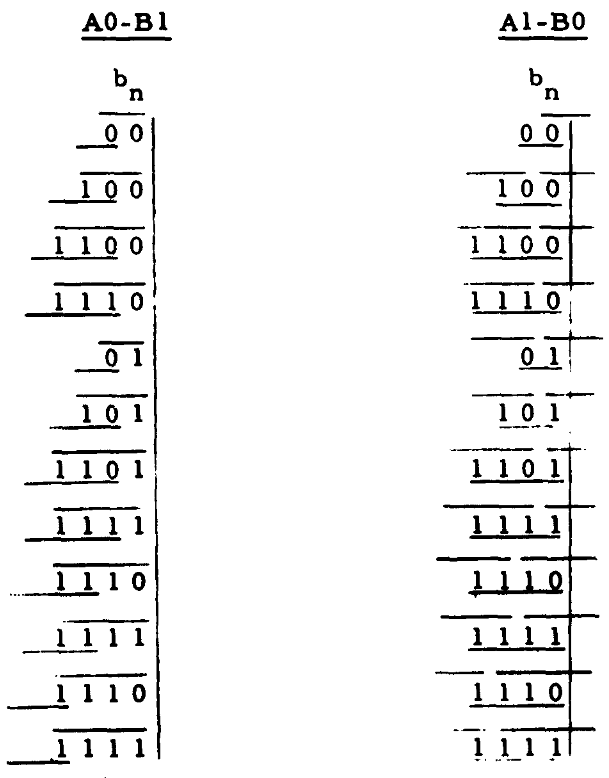


FIGURE 24. SEQUENCES WITH A0-B1 AND A1-B0 CONFIGURATIONS.

It can be readily demonstrated that affixing the same code will force synchronization for every sequence in which the B register contains a given terminator irrespective of the codes whose partition resulted in that terminator. Thus the first four A0-B1 sequences of Figure 24 can be brought to synchronization by adding the code 100 as shown below.

In analyzing the A1-B0 sequence, the presence of binary digits in the A register at b_n indicates that the code has not terminated at b_n. Our concern is with that part of the code which appears at or before b_n, however, rather than with the part that follows b_n. The code may be considered to be partitioned at b_n, the digits at or before b_n forming a prefix, the digits after b_n forming a terminator which is discarded. Analysis shows that treatment of this sequence is the same as for the A0-B1 sequence since A0-B1 and A1-B0 are conjugate pairs, and the same code that is used to force synchronization with the A0-B1 configuration can also be used with the A1-B0 configuration as shown below. Thus the eight A0-B1 and A1-B0 sequences whose prefix or terminator is 0 can all be forced into synchronization by affixing the code 100.

A0-B1

0 0	1 0 0
1 0 0	1 0 0
1 1 0 0	1 0 0
1 1 1 0	1 0 0

A1-B0

0 0	1 0 0
1 0 0	1 0 0
1 1 0 0	1 0 0
1 1 1 0	1 0 0

Further analysis shows that the eight sequences whose prefix or terminator is 1 can be forced into synchronization by affixing the code 00. The remaining A0-B1 and A1-B0 sequences can be brought into synchronization by affixing codes as shown in Figure 25.

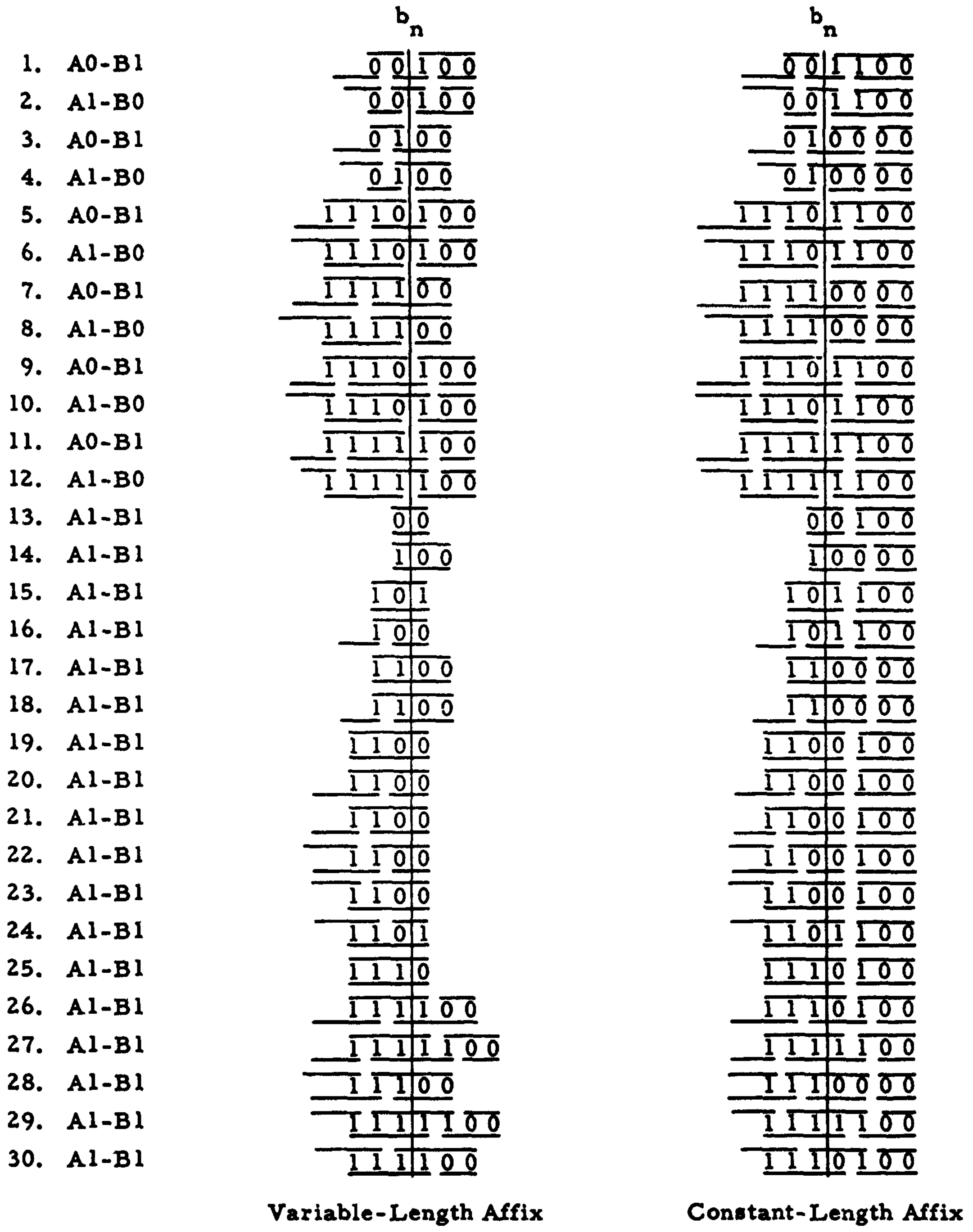


FIGURE 25. FORCED SYNCHRONIZATION OF ALL SEQUENCES IN A0-B1, A1-B0, AND A1-B1 CONFIGURATIONS WITH VARIABLE AND CONSTANT LENGTH AFFIXES.

The A1-B1 configuration is more difficult to analyze. The 18 possible sequences of this configuration are shown in numbers 13 - 30 of Figure 25. The digits to be affixed to an A1-B1 configuration to force synchronization need not comprise a code. The sequence $\underline{1}|\overline{0\ \ }$, for example, can be brought to state S by affixing one digit, 0, which is not a code. All the A1-B1 sequences can be forced into synchronization by affixing from 1 to 4 digits as shown in Figure 25.

The left-hand column of this figure shows the variable-length affixes required to force synchronization at b_n. By applying the analysis of partitions of the encoding to each sequence and selecting binary digits that meet the requirements of synchronization constant-length affixes can be determined as shown in the right-hand column of Figure 25.

Forcing synchronization of the string of binary digits in θ_r, when required, is now a straightforward matter. The message will always terminate in an end of message (EOM) code. Since the length of a synchronizing sequence is not fixed, the complete EOM code will always appear in θ_r and hence the configuration of the A and B registers will either be A0-B0 or A0-B1. The former is the synchronized state and requires no special treatment. The contents of the B register in the latter configuration will be one of the n-1 ways the EOM code can be partitioned into two strings. Some of the partitions will produce C-terminators, in which case synchronization will occur. Other partitions will leave a string of binary digits which form a C-prefix. Addition of an appropriate C-terminator to the C-prefix completes the synchronizing sequence.

At machine R, the affixed digits following an EOM code are discarded after the error checking process is completed.

The redundancy of the free-running system is $R = \frac{k}{m+k}$, where k is the number of check digits and m is the number of information digits in a synchronizing sequence. The value of m is the average length of a synchronizing sequence, $\bar{\sigma}$, as determined for a particular encoding. The value of k is the sum of the digits in the SFC and the CHECKWORD. The digits affixed to force synchronization at EOM can be ignored as their number in relation to the total message digits will be very small. The redundancy is, therefore,

$$R = \frac{\text{SFC + CHECKWORD}}{\bar{\sigma} + \text{SFC + CHECKWORD}} .$$

The error detection capabilities and the probability of undetected error of the free-running self-synchronization system are examined in the following section.

E. Experimental Results

1. Length of Synchronizing Sequences

While it is evident from Figures 17 and 18 that a sequence of codes that is never self-synchronizing can be generated, a message formed from symbols whose frequency distribution is that expected in a natural language will always self-synchronize. Self-synchronization is assured because the shorter codes will appear more often than the longer codes. The shorter codes will also frequently appear imbedded in the longer codes which are formed by concatenating the shorter codes with different prefixes. The presence of C-terminators and multiple C-terminators are preconditions of self-synchronization.

The self-synchronizing properties of the non-spaced minimum redundancy encoding of the 5,208 codes described in this report have been analyzed. Approximately 5,000 synchronizing sequences were obtained from a variety of messages. The lengths of the self-synchronizing sequences were determined by processing the messages through the free-running system and printing out the number of binary digits in a synchronizing sequence with the initial state of the machine set at F (minus one digit). The more than 4,000 synchronizing sequences contained in a 19,710 word message ranged in length from 7 to 658 binary digits. The first 2,000 sequences contained 148,379 binary digits for an average length of 74.2 digits per synchronizing sequence. The median length was 52.6 digits. The distribution of the lengths is shown in Figure 26.

Three smaller samples contained nearly 1,000 synchronizing sequences with lengths ranging from 7 to 488 binary digits. The average length of a synchronizing sequence in these samples ranged from 72.7 to 76.5 binary digits per sequence.

Typical lengths (in binary digits) of synchronizing sequences as they were found in a message are:

89 : 16 : 43 : 247 : 23 : 164 : 43 : 15 : 15 : 22 : 126 : 28 : 31 : 81: . . .

Sample sequences are:

(89) GOOD $\overset{\frown}{\text{E S}}$ → GOOD TIME S OR BAD - DESPITE TALK OF

(16) SLUMP

(43) , THE ECONOMY IS STRONG

(247) IN MANY KEY % S· E· C· T· O· R· S▵- % G· N· P▵, INDUSTRIAL
PRODUCTION, PERSONAL INCOME, BUILDING ARE
SET TING RECORD PACE

(23) - CONCERN

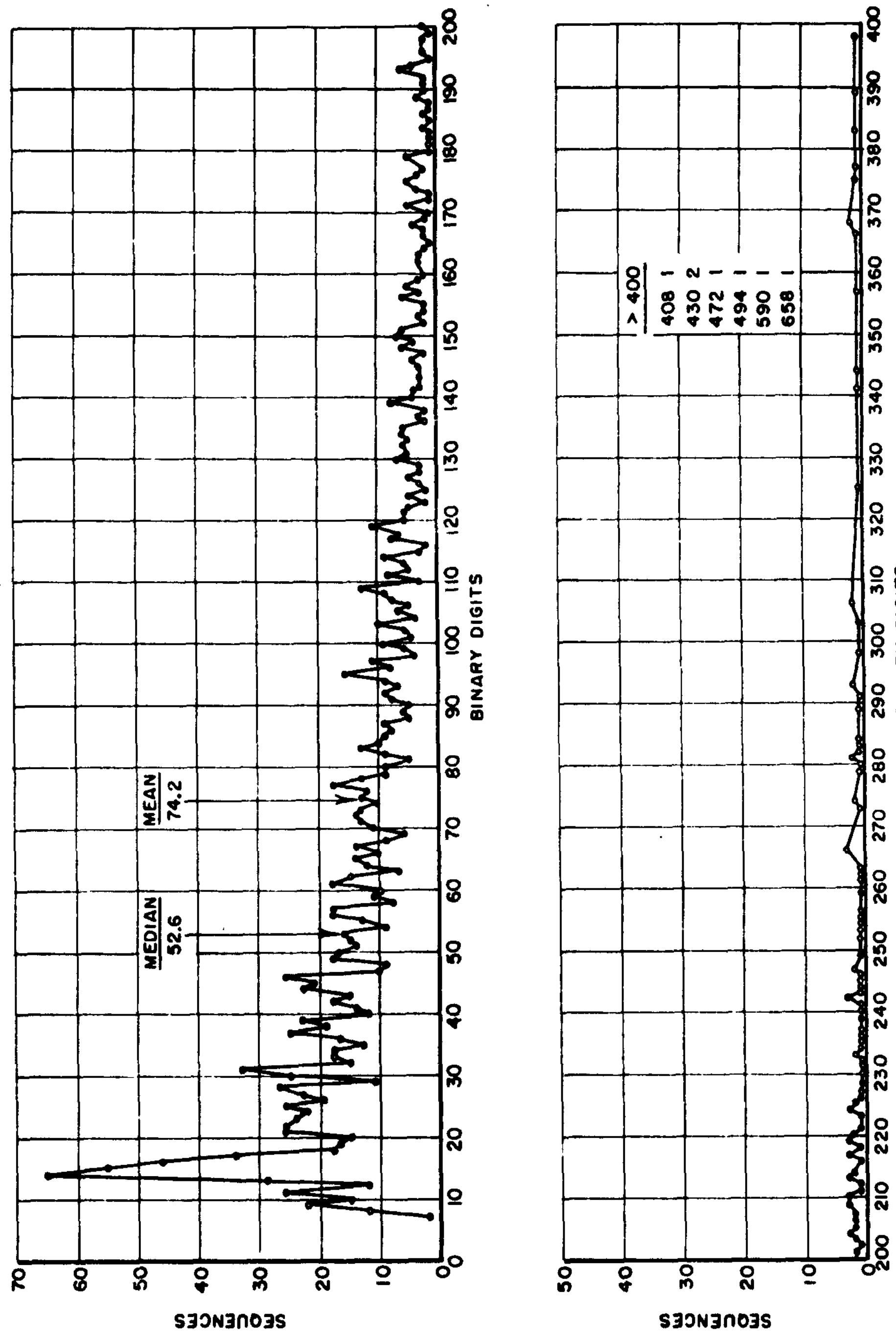


FIG. 26 LENGTHS OF 2000 SELF-SYNCHRONIZING SEQUENCES OF 5208-CODE ENCODING

(Spelled words are indicated with the symbol · between letters; the circumflex denotes synthesis; % and △ are letter mode entry and exit operators).

The reasons for the relatively short length of a synchronizing sequence are apparent from an analysis of the 5,208 codes. The first three codes of the encoding, for example, are 0000, 0001, and 0010, and appear as C-terminators in 322, 322, and 323 codes respectively, for a total of 967 codes. Multiple C-terminators of eight binary digits consisting of any two of the three codes in juxtaposition appear in 180 codes.

A number of codes generated in the canonical encoding contain compound C-terminators, that is, terminators that are associated with more than one prefix set. Thus a 14-digit code is a C-terminator for both a 15-digit and 16-digit code as shown in partitions 1-3 of Figure 28. Similarly, a 13-digit code is a C-terminator of a 14-digit and 15-digit code:

```
11· 1101111010001        (15)
 1· 1101111010001        (14)
    1101111010001        (13)
```

The 10-digit codes from 101· 0101000 to 101· 0110101, as another example, contain C-terminators that comprise the complete set of fourteen 7-digit codes, etc.

Of particular interest is the distribution of the lengths of self-synchronizing sequences of Figure 26 plotted on probability paper. It is noted in Figure 27 that the probability a synchronizing sequence will exceed 100 binary digits in length is 0.215; the probability that 200 binary digits will be exceeded is 0.042; while the probability that the length will be greater than 300 binary digits is only 0.0035.

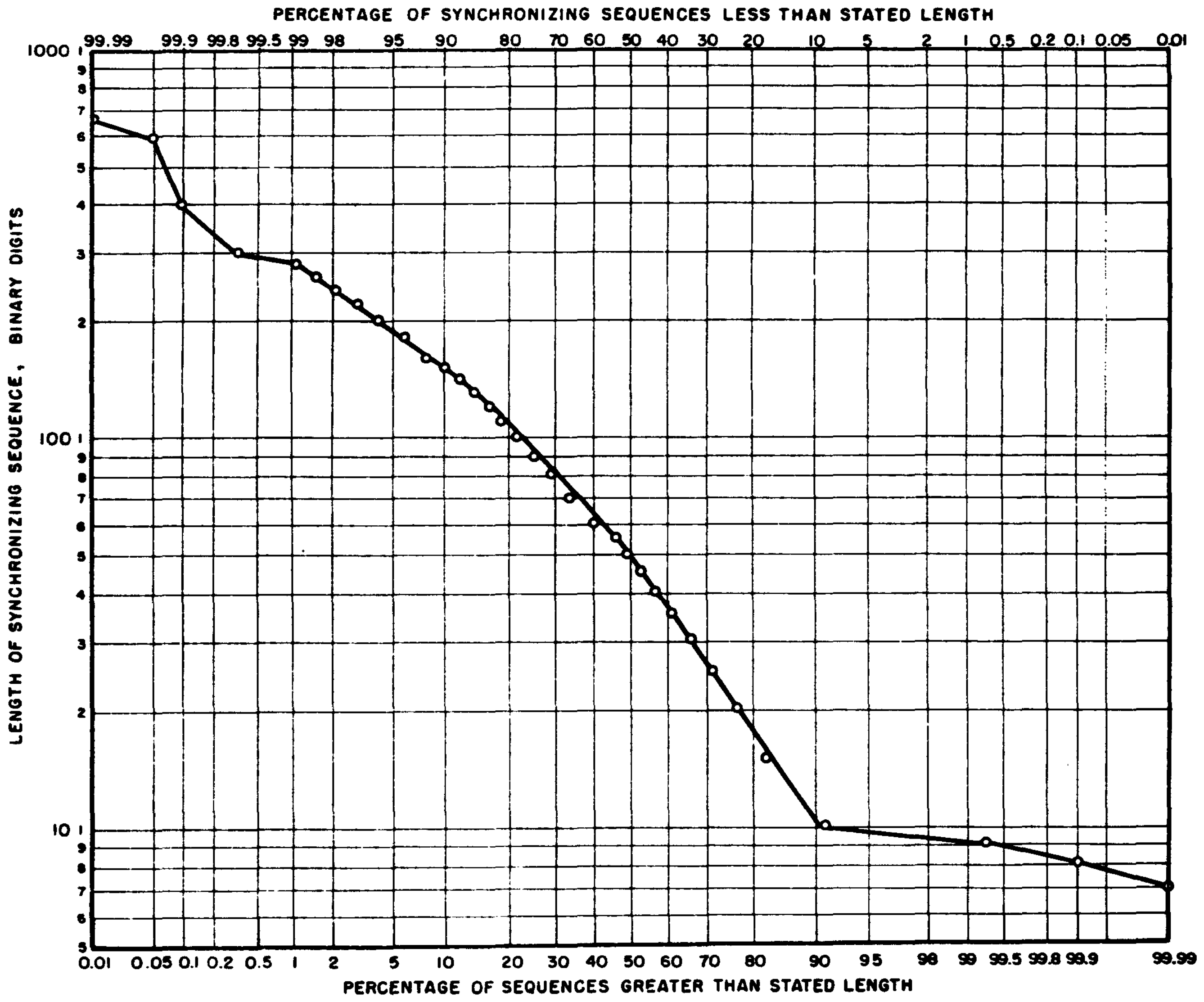


FIG. 27 PROBABILITY DISTRIBUTION OF LENGTHS OF SELF-SYNCHRONIZING SEQUENCES

2. Forced Synchronization at End of Message

The EOM code in the experimental system is a 16-digit code which can be partitioned into 15 different pairs of disjoint subsets as shown in Figure 28. Since the A register will be empty after the EOM code is deciphered, one of the 15 partitions will appear in the B register if the terminal sequence does not self-synchronize. The first three partitions will result in self-synchronization after the B register is deciphered. The remaining 11 partitions, after deciphering, have five classes of remainders as shown at the bottom of the figure. These remainders can be forced into synchronization by affixing one of two C-terminators.

3. Error Detection Capabilities

An error will be detected if any of the checks that were previously listed fail. Error detection is certain for the following types and location of errors:

1. Sync frame code
   a. Initial
      (1) Transposition error(s) - any configuration
      (2) Deletion of digits that cannot be replaced by borrowing from succeeding synchronizing sequence

         (e. g. , SFC = 00110:

~~0~~0110· X	detected
~~00~~110· XX	detected
~~001~~10· XXX	detected
0~~011~~0· XXXX	?)

PARTITIONS OF EOM CODE

1. $\underline{\quad 1}\cdot\underline{1\,1\,1\,1\,0\,1\,0\,0\,1\,0\,0\,0\,1\,0\,0}\!\downarrow_{S}$
2. $\underline{\quad 1\,1}\cdot\underline{1\,1\,1\,0\,1\,0\,0\,1\,0\,0\,0\,1\,0\,0}\!\downarrow_{S}$
3. $\underline{\quad 1\,1\,1}\cdot\underline{1\,1\,0\,1\,0\,0\,1\,0\,0\,0\,1\,0\,0}\!\downarrow_{S}$
4. $\underline{\quad 1\,1\,1\,1}\cdot\underline{1\,0\,1\,0\,0\,1\,0\,0\,0\,1}\;\underline{0\,0\quad}$
5. $\underline{\quad 1\,1\,1\,1\,1}\cdot\underline{0\,1\,0\,0\,1\,0}\;\underline{0\,0\,1\,0}\;\underline{0\quad}$
6. $\underline{\quad 1\,1\,1\,1\,1\,0}\cdot\underline{1\,0\,0\,1\,0\,0\,0\,1\,0}\;\underline{0\quad}$
7. $\underline{\quad 1\,1\,1\,1\,1\,0\,1}\cdot\underline{0\,0\,1\,0}\;\underline{0\,0\,1\,0}\;\underline{0\quad}$
8. $\underline{\quad 1\,1\,1\,1\,1\,0\,1\,0}\cdot\underline{0\,1\,0\,0\,0\,1}\;\underline{0\,0\quad}$
9. $\underline{\quad 1\,1\,1\,1\,1\,0\,1\,0\,0}\cdot\underline{1\,0\,0\,0\,1\,0\,0\quad}$
10. $\underline{\quad 1\,1\,1\,1\,1\,0\,1\,0\,0\,1}\cdot\underline{0\,0\,0\,1}\;\underline{0\,0\quad}$
11. $\underline{\quad 1\,1\,1\,1\,1\,0\,1\,0\,0\,1\,0}\cdot\underline{0\,0\,1\,0}\;\underline{0\quad}$
12. $\underline{\quad 1\,1\,1\,1\,1\,0\,1\,0\,0\,1\,0\,0}\cdot\underline{0\,1\,0\,0\quad}$
13. $\underline{\quad 1\,1\,1\,1\,1\,0\,1\,0\,0\,1\,0\,0\,0}\cdot\underline{1\,0\,0\quad}$
14. $\underline{\quad 1\,1\,1\,1\,1\,0\,1\,0\,0\,1\,0\,0\,0\,1}\cdot\underline{0\,0\quad}$
15. $\underline{\quad 1\,1\,1\,1\,1\,0\,1\,0\,0\,1\,0\,0\,0\,1\,0}\cdot\underline{0\quad}$

AFFIXED C-TERMINATORS

a. $\overline{0\cdot\underline{1\,0\,0\,1\,0}}\,\underline{\overline{0\,0\,0\,0}}$ $\qquad C_A$

b. $\overline{0\,0\cdot\underline{0\,1}}\,\underline{\overline{0\,0\,0\,0}}$ $\qquad C_B$

c. $\overline{1\,0\,0\cdot\underline{0\,1\,0}}\,\underline{\overline{0\,0\,0}}$ $\qquad C_B$

d. $\overline{0\,1\,0\,0\cdot\underline{0\,1}}\,\underline{\overline{0\,0\,0\,0}}$ $\qquad C_B$

e. $\overline{1\,0\,0\,0\,1\,0\,0\cdot\underline{0\,1}}\,\underline{\overline{0\,0\,0\,0}}$ $\qquad C_B$

FIGURE 28. FORCED SYNCHRONIZATION OF EOM CODE

(3) Insertion error(s) that will not shift digits to form valid SFC

(e.g., SFC = 00110:

```
↓
100110      detected
↓
000110      detected
 ↓
010110      detected

        etc.)
```

b. Terminal (no intervening errors)

(1) Transposition error(s) - any configuration

(2) Deletion error(s) - same as for initial SFC

(3) Insertion error(s) - same as for initial SFC

2. CHECKWORD (no errors in synchronizing sequence)

a. Transposition error(s) - any configuration.

The probability of detecting errors other than those described above and all types of errors in a synchronizing sequence are dependent upon the synchronizing sequence in which the errors appear and upon the error configuration. To be undetected an error must generate a synchronizing sequence that is followed by a terminal SFC and meets all the checks of either the original CHECKWORD or a falsely generated CHECKWORD.

Although the probability of every code is known, a mathematical determination of the probability of undetected error would require a knowledge of the probability of a given concatenations of codes, the probability that an error will occur at a given location in this concatenation of codes, the probability that the concatenation of codes will be transformed into another concatenation of codes, and the probability that the resultant concatenation will fail one of the error checks. Accordingly, estimates of the probability of detected and undetected error were derived from an analysis of a large number of messages into which errors had been inserted.

The errors were generated by deleting and/or transposing (changing 0 to 1 and conversely) digits in designated locations in single and multiple error configurations. Errors were inserted in the first synchronizing sequence of the message which ranged in length from 14 to 378 binary digits. Five test series were analyzed.

From one to six consecutive digits were deleted starting at the CHECKWORD in one test series thus generating a false CHECKWORD since a specified number of digits following an SFC code are accepted as a CHECKWORD by the error detection program. One to ten consecutive digits were deleted beginning at the first digit of the first synchronizing sequence of the messages in another series. From one to five consecutive digits were deleted within the first synchronizing sequence in a third test series. These three test series checked 660 messages under circumstances simulating loss of synchronism and burst errors.

Two digits were transposed in the initial synchronizing sequence of messages of the fourth test series and two digits were transposed and one deleted in the fifth series. A total of 1,060 messages were checked for errors and the results are tabulated in Table 23.

A 5-digit sync frame code (00110) was used in the tests and the CHECKWORD consisted of six binary digits, utilized as follows:

$b_1 \cdot b_2 \cdot b_3 \cdot b_4$	= number digits in synchronizing sequence (mod 16)
b_5	= number 0's in synchronizing sequence (mod 2)
b_6	= number 1's in synchronizing sequence (mod 2).

TABLE 23

ERROR DETECTION ANALYSIS

ERROR CONFIGURATION	NO. MESSAGES	σ_1											σ_2										$> \sigma_2$					UNDETECTED ERROR
		< SFC						@ SFC					< SFC					@ SFC										
		1	2	3	4	5	6	1	3	4	5	6	1	3	4	5	6	1	3	4	5	6	1	3	4	5	6	
Delete 1-6 digits in CHECKWORD	120		75	17			5		10			6		3					2					1				1 @ SFC σ_1
Delete 1-10 initial digits of σ_1	200	2		76	3		23		39			21	2	17	1				9	1		1		3				2 @ SFC σ_2
Delete 1-5 digits in σ_1	340	3		108	10		67		60			30	3	16	9		1		21					11			1	0
																		$> \sigma_1$										
																		1	3	4	5	6	7					
Transpose 2 digits in σ_1	200	2		82	10		11			19		10						1	37	1		2						25 @ SFC σ_1
Transpose 2 digits and delete 1 digit in σ_1	200	7		83	3		20		22			18							38			6	1					2 @ SFC σ_1
Total	1,060	14	75	366	26		126		131	19		85																30

The redundancy of the error detection system was

$$R = \frac{6 + 5}{74 + (6+5)} = 0.13.$$

The following error types are designated:

1. SFC not present
2. $\left[b_5 + b_6\right]$ (mod 2) $\neq b_4$ of received CHECKWORD
3. Digit count of received synchronizing sequence (mod 16) $\neq$ $b_1 \cdot b_2 \cdot b_3 \cdot b_4$ of CHECKWORD
4. 0 count (mod 2) of received synchronizing sequence $\neq b_5$ of received CHECKWORD
5. 1 count (mod 2) of received synchronizing sequence $\neq b_6$ of received CHECKWORD
6. State $\overline{S}$ followed by SFC
7. State $\overline{S}$ at EOM.

Errors were detected in the order shown. In the event a message failed several checks, only the first one that was encountered was recorded. If a synchronizing sequence was obtained and checks 1 through 7 were successful, check 1 was repeated to determine the presence of an initial SFC for the next succeeding synchronizing sequence. This SFC also served as the terminal SFC of the preceding sequence.

The position in the digit string at which an error was detected is also indicated in Table 23. Since the number of binary digits in the correct sequences was known, this position was readily determined from the error printout which indicated the number of binary digits in the error synchronizing sequence.

An estimate of the probability of occurrence of the pseudo-isomorphic condition together with the probability of over-running a terminal SFC can be made using the figures of the Table. Synchronizing sequences which terminated at the original SFC for any σ_i exhibit the property of pseudo-isomorphism. In these cases, error detection was wholly dependent upon the CHECKWORD. In other cases, the self-synchronizing property can be said to have made possible the detection of error.

Error detection occurred within the first synchronizing sequence before the terminal SFC in 57.2 percent of the test messages and at the terminal SFC of σ_1 in 20.3 percent of the messages. In 19.7 percent of the messages, the terminal SFC of σ_1 and the CHECKWORD of σ_2 were over-run. The maximum number of synchronizing sequences deciphered before discovery of error was four and the total number of digits in the transformed sequence was never greater than 300. These results agree with the probability curve of Figure 27.

It is noted that an SFC code was deciphered within σ_1 in 11.9 percent of the messages and an EOM code was falsely generated within a message in one message. Error type 6 at the terminal SFC of σ_1 which occurred in 85 messages indicated that the A registers of the correct and error messages were in synchronism at the SFC but the B registers were not in synchronism.

In all, 77 percent of the errors were detected by the mechanism of self-synchronism and 23 percent exhibited the property of pseudo-isomorphism and depended wholly upon the CHECKWORD for error detection.

The effects of deletion and transposition errors upon messages are different. Deletion errors will always result in a change of length of some codes whereas transposition errors need not necessarily change the length of any code. The latter case occurs when a digit is transposed in the β-portion of a code whose prefix is unaffected, and the changed β-portion does not transform the code to a different length code. In the 16-digit codes of the 5,208-code encoding, for example, 2^9 codes have the same prefix and hence transposition of from one to nine terminal digits would have no effect on the code length. This holds true only for the A register, however, as a different partition of a code with transposed digits can occur in the B register.

When the CHECKWORD was mutilated by deleting digits, 62.5 percent of the errors were detected by the zero and one check (error type 2) on the generated CHECKWORD. The 19 errors detected by the failure of the zero digit count in double transposition errors illustrates the value of this check.

In 50 messages, a synchronizing sequence was obtained and all checks on the CHECKWORD were satisfied. Twenty of these sequences occurred prior to a normally situated SFC and error was detected by the absence of an SFC following self-synchronization. In 30 messages all checks were successful and errors were undetected. In every instance of undetected error, the pseudo-isomorphic condition was present. No case was found of an undetected error occurring within a correct synchronizing sequence. Only two of the undetected errors included two synchronizing sequences, all others being confined to one sequence. In general, the length over which an undetected error can be propagated will be determined by the probability curve of sequence lengths.

The test results indicate that undetected errors occur with greatest probability under the condition of pseudo-isomorphism. For the case of pseudo-isomorphism occurring within one synchronizing sequence, the following types of errors will always be detected:

1. Deletion of any number of digits less than the modulus m of the total digit count
2. Any odd number of transposition errors
3. Any even number of transposition errors, provided the number of 0's or 1's transposed are odd.

As with conventional error detection codes, the probability of undetected error is a function of the redundancy added to the information digits. With the addition of more redundant digits, the probability of undetected error in the self-synchronizing system can be made vanishingly small. The addition of three additional digits to the CHECKWORD, it is estimated, would make possible the detection of the errors that were undetected in the test series. A digit count (mod 32) and zero and one counts (mod 4) would impose additional checks on the CHECKWORD itself and would decrease the probability that an error string of digits would meet all checks simultaneously. Other methods of checking the pseudo-isomorphic condition can also be considered.

Designation of the serial number of a synchronizing sequence can be effective in detecting the presence of undetected error which over-runs one synchronizing sequence. A serial number would also be effective in detecting the deletion of a complete synchronizing sequence or several sequences arising from a prolonged error burst as encountered under fading conditions. The serial number would also facilitate error correction through a feedback channel by identifying synchronizing sequences that are in error and that are retransmitted.

F. The Fixed-Length Sequence

Because the SFC and CHECKWORD are determined on the basis of frequency and reliability requirements, and the average length of a synchronizing sequence is determined by the structure functions of an encoding, there is little flexibility in the free-running system in varying the redundancy introduced by the error checks. Accordingly, the possibility of extending the system to a fixed-length sequence was studied.

Three major problems arise in the design of a fixed-length self-synchronizing error detection system. First, there is the problem that a string of binary digits will self-synchronize before the n-th digit, where n is the assigned length. Because the sequential machine is loosely connected, once it enters state S it cannot leave this state. It is easy to see that any error that follows the S state will be undetected since by the exhaustive property of the encoding the succeeding digits will still be uniquely decipherable and the deciphering process will generate the same code simultaneously in both A and B registers. This problem is readily met by returning the machine to the initial state q_0 every time it enters state S. In this manner, as the deciphering of the string of binary digits takes place a series of synchronizing sequences can be formed until the n-th digit is reached.

The second problem arises from the configuration of the A and B registers at the n-th digit. Although it was demonstrated that synchronization could be forced for the A0-B1, A1-B0, and A1-B1 configurations, given an encoding consisting of thousands of codes with the code lengths required in a large encoding, it is necessary to formulate a procedure for forcing synchronization. The procedure must be capable of dealing with the very large number of permutations that can be found by partitioning a long code in all possible ways and distributing these partitions especially in the A1-B1 configuration. It must also determine the minimum number of added digits required by every possible configuration for every code.

The A1-B1 configuration can be eliminated from consideration by adopting the principle that the A register must always be zero at the time of error checking. Since the A0-B0 configuration is the S state, that leaves only the A0-B1 configuration. (It was previously noted that the A1-B0 configuration is a conjugate of the A0-B1 configuration.) The A register, however, will not necessarily be empty at b_n. To empty the A register it is necessary to complete any code that has been started before the n-th binary digit. The number of digits that will follow b_n will therefore vary, the bounds on the number being 0 to $L_{max}-1$, where L_{max} is the code of maximum length in an encoding. The average value of the number of binary digits that will appear after b_n will be considerably less than $\frac{L_{max}-1}{2}$ since the codes are not equiprobable.

Because the length of the check sequence will vary between n and n+c, where c is the number of digits in that portion of a code which is completed after b_n, the system will be quasi-fixed length. Although the value of c is dependent on the message sequence, its upper bound is known and its length will be small compared to the value of n which can be set at any desired value.

The third problem is concerned with forcing synchronization if the machine is not in the S state at b_n or b_{n+c}. In the earlier discussion it was shown that the A0-B1 configuration can be forced into synchronization by considering the contents of the B register at b_n or b_{n+c} as a C-prefix and affixing a C-terminator. Since the C-prefix determines the characteristics of the C-terminator which is affixed, it is the C-prefix alone that contains the information as to the status of the message sequence at the (n+c)-th digit. By placing the contents of the B register, designated BREG, in a predesignated position of the check sequence, a representation is obtained of the status of the message at the (n+c)-th digit. The C-terminator can be entirely eliminated and no calculation is required to determine a minimum configuration of added digits. The contents of the B register can be obtained from the deciphering routine in the normal manner.

The contents of the B register at the (n+c)-th binary digit can be placed directly following the (n+c)-th digit. These digits will not form a code for otherwise they would have been eliminated from the B register. They do not require special framing, however, as does the CHECKWORD, since they will always appear after the (n+c)-th binary digit. Because n is known, the code begun in the A register at the n-th binary digit is known, and the value of c is determined by this code. The position of the contents of the B register are, therefore, framed by the (n+c)-th binary digit and the terminal SFC.

The longest remainder that can appear in the B register is $L_{max}-1$ since it is possible for the entire c portion of the synchronizing sequence to remain in the B register. The average value of the number of digits that will appear in the BREG position will again be less than $\frac{L_{max}-1}{2}$.

It appears that the three problems confronting the design of a quasi-fixed length self-synchronizing error detection system have been solved. In reality, however, the quasi-fixed length system reduces to little more than a parity check over n+c binary digits with relegation of the self-synchronizing property to but a very small role in error detection. This seeming contradiction arises from the phenomenon of pseudo-isomorphism coupled with the probability distribution of the length of self-synchronizing sequences. In short, the ability of a string of binary digits to self-synchronize after any mutilation by errors and the very high probability that it will return to synchronism within a small range of sequence lengths, hinders extension of the error detection system beyond the free-running system.

Consider, for example, a fixed length sequence of 200 binary digits containing the following self-synchronizing sequences: 40, 20, 80, 15, and 310 binary digits in that order. The last sequence of 310 digits would be split at the 45th (or 45 + c)-th digit to complete the n = 200 requirement.

The probability that a self-synchronizing sequence will not return to synchronization by the 40th digit is 0.56 (from Figure 27). If an error occurs in σ_1 and overruns into σ_2, the probability that a sequence will not have returned to synchronism by the 60th digit is 0.40. If the error also overruns σ_2 and continues into σ_3, the probability that self-synchronization has not occurred by the 140th digit is only 0.12. At the end of σ_4, the probability has further decreased to 0.09. Thus if any configuration of errors transforms the sequence of 200 digits in any way, the probability that it will not have returned to synchronization by the 155th digit is only 0.09. The efficacy of the forced synchronization or B register check on the status of the 200 digit sequence is, therefore, very small.

At the 200th digit, the probability of non-return to synchronism becomes increasingly small, namely, 0.04. By the 300th digit the probability is further reduced to 0.0035.

It is apparent that if the number of digits erroneously deleted is greater than c + BREG, the requirement that n = 200 will cause the terminal SFC and perhaps some portion of the following n-digit sequence to be considered as part of the first 200-digit sequence. In this case again, the probability that the sequence will have returned to synchronism before the c + BREG region is very high and consequently, the deletion errors will have the effect of a left shift of the digits following synchronism until the n-th digit (including borrowed digits) is reached. Thus it is the digit count rather than the self-synchronizing property that contributes most to error detection in a fixed length system and the error detection capabilities of the system are greatly reduced.

IX. COMPUTER PROGRAMS

The computer programs used in the adaptive information transmission system are described in this section. Although programming was done for the UNIVAC 1105 computer, the flow charts have been prepared as machine-independent and the program descriptions accompanying the flow charts are of a similar nature.

Charts 1 through 6 are programs for dictionary preparation, encoding, and statistical and linguistic analysis. They describe in detail the generalized system charts shown in Figures 1 and 2.

Charts in the 7. series are subroutines used in the transmitter program for enciphering messages as depicted in simplified form in Figure 3. Charts 7. 1 through 7. 13 are the routines used for preparing messages for transmission without error checking. If error checking is employed, using the error detection system described in Section VIII, routines of flow charts 7. 14 through 7. 18 are substituted for the output routine of chart 7. 13.

Charts in the 8. series are detailed flow charts of the general program for deciphering and decoding a message at the receiver as shown in Figure 4.

Chart 9 is an alternate program that can be used at the transmitter. The system employs magnetic tapes with pre-sort and post-sort routines. This system has not been programmed or tested.

Tags and names of subroutines appear in upper case in the program descriptions. Tags are used to identify connectors in the flow charts. Subscripted X's are used to designate connectors employed because of space limitations on the charts. Subroutines are designated by hexagonal boxes.

A. Dictionary Tapes, Encoding, Analysis

Preparation of Transmitter and Receiver Dictionary Tapes (Chart 1)

Transmitter and receiver tapes used in the encoder and decoder operations are prepared by a sequence of routines which accept raw input data and convert these data into the required formats. A general flow chart of the sequence of computer routines and the eight basic tapes that are generated is shown in Chart 1. Tapes T-1 through T-6 are intermediate tapes. Tapes T-6 and T-7 are dictionary operations tapes used in the decoder at the receiver terminal; T-8 is the dictionary operations tape used in the encoder at the transmitter terminal.

The original dictionary Source Tape was produced from a library of IBM cards via card-to-magnetic-tape converter which allocated one blockette on the tape to each card. Data in blockettes were designed in the following form:

1. Two computer words equivalent to 12 XS-3-coded characters were reserved for one dictionary word; words in excess of 12 letters were eliminated.

2. One computer word was allowed for the frequency count;

3. One computer word was allocated for the number of letters in the word listed;

4. One computer word was allowed for the synthesis symbol (SYNTAG).

Before card-to-tape conversion, all cards were sorted in order of increasing word frequency.

Word list and rank-frequency tapes, T-1, T-2

Tapes T-1 and T-2 are prepared by one program which generates the 4-computer word format shown in T-1 from the card-converted format.

At the same time, rank numbers are assigned to each word and associated with a frequency as shown in T-2. Ties are ignored and rank numbers are assigned consecutively.

Frequency tree generation, T-3 (Chart 2)

The object of this program is to generate the Huffman frequency tree using the bottom merge for combined frequencies. The program begins with reading of control data and the rank-frequency tape T-2. Initialization of the storage allocation parameters follows.

The two smallest frequencies, F_i and F_{i+1} are added to form COMB. The second of the next pair of frequencies is placed in STORE. End of T-2 data decision determines whether the program jumps to the MERGE sequence or, when negative, evaluates the magnitude of STORE and COMB. This test permits a number of combinations in one pass without violating the rule of always merging the two smallest remaining frequencies.

If the value of STORE is equal to or greater than COMB, the MERGE sequence is followed. Otherwise, the value of COMB is placed in temporary storage and the first combined rank R_k^* is generated. The starred rank is designated by adding the value of k to a large constant number (bit 36 = 1).

When all possible combinations in a pass have been made, the MERGE sequence merges the combined (R_k^*) frequencies in their proper locations in the reduced rank-frequency T-2 table. If R_k^* is equal to or greater than the first frequency in the T-2 table, the table location is incremented, and a comparison is made with the next frequency. This continues until the combined frequency is merged. When the combined frequency equals the table frequency, the combined frequency is merged after equal uncombined frequencies. A check is made to ensure that all combined frequencies have been merged before the COMBINE sequence is re-entered.

The program terminates when j = N - 1 indicating that all combinations have been generated. The final value of COMB is the cumulative frequency of all entries in the dictionary.

Determination of Code Breakpoints, T-4 (Chart 3)

The output of tape T-3 is a table listing all nodes of the frequency tree in ascending order from 1 to N-1. Associated with each node are the two frequencies that were combined to form that node. The code breakpoint routine determines the number of codes to be assigned each code length and the maximum code length in the encoding.

This program divides Tape T-3 which is the node table illustrated in Figure 10, into a number of segments and searches each segment for an uncombined rank or blank node. R^* designates a proper node formed from a combination of two frequencies.

The program starts by initializing $M_1 = M_2 = N-1$ and $i = 1$. M is the current node (rank) being examined, M_2 is the upper boundary (greatest rank) of a segment and M_1 is the lower boundary (smallest rank) of a segment.

Location of the first blank node or unstarred R establishes the boundary for codes of length i binary digits. If a blank node is not found in either R_1 or R_2, a null set is indicated and the R^* designates the boundary of the next segment of the table to be searched.

The previous lower boundary M_1 is substituted for the previous upper boundary M_2, i the code length is incremented, and the search for a blank node begins again with M_1 at location $\textcircled{1}$. The entire interval between M_1 and M_2 is searched by successively incrementing M, the current rank, until $M = M_2$.

The program terminates when the final segment of the node table has been searched. A new table, tape T-4 is then printed out.

<u>Code Assignment, T-5</u> (Chart 4)

The code assignment program generates a canonical encoding using as input the code breakpoint table contained in tape T-4.

The program begins with reading of T-4 and initializing i, the code length, j, the number of codes generated in code length i, and k, the serial number of the generated codes. Encoding starts with an initial code of zero. A binary one is inserted before the code to serve as a marker for the leading zeros.

A check is first made to determine whether all codes have been assigned, a condition satisfied when $k = N$, where N is the number of codes in the encoding. If not, the number of codes of length i as determined by the breakpoint table is entered into n. A null set in which $n = 0$ leads to shifting left one position (equivalent to adding a zero) and incrementing i.

If a set of codes of length i which is not null is encountered, the previously generated code is assigned to the k-th code and this initial code is incremented by binary one until $n = j$. When all codes of a given length have been generated, the last code is shifted one position to the left and becomes the first of the new set of codes of length $i + 1$.

<u>Code Breakpoint Table, T-6</u> (Chart 5)

The code breakpoint table program is designed to prepare tape T-6 for incorporation into the decoder program. The table is the basis for deciphering the received bit string.

The program begins with normal start, reading of control data, and reading of magnetic tape containing the codes. Storage space allocation is initialized with CODE (referring to code storage), DATA (referring to the summary table), and SCREEN - code masking constant. Decision is made if all the codes have been examined at Γ_1. If the decision is yes, DATA is dumped on magnetic tape and the computer stops; if not, the code is transfered to an examination location, EXAM, and masked with SCREEN value. When EXAM value is not equal to zero, the program jumps to Γ_2, which deals with the last code of a given code length sequence. If EXAM is equal to zero, the analysis is continued in the same code length sequence. The value of Flag 1 controls the listing of the analysed code in the DATA table. If Flag 1 is not equal to one, Flag 1 is set and the code under consideration is entered in the data table. CODE is then incremented and program jumps to Γ_1. If Flag 1 is equal to one, the current code value is transferred to an appropriate DATA location, CODE is incremented, and the jump to Γ_1 performed.

At the Γ_2 branch two cases are handled: (a) if Flag 1 is set equal to one, a given code length sequence is terminated with insertion of the last code value. Flag 1 is then set to zero, DATA and SCREEN are incremented for next code length, and the program jumps to Γ_1. If Flag 1 is not equal to one, three consecutive DATA positions are set equal to zero which indicate a vacuous code group. DATA and SCREEN are then incremented and the program is continued at Γ_1.

<u>Receiver and Transmitter Dictionary Tapes, T-7, T-8</u>

The T-7 and T-8 tapes are the dictionary tapes used at the receiver and transmitter terminals respectively. Their preparations entail use of a number of computer programs which include standard data processing routines.

Tape T-7 is prepared by merging tapes T-1 and T-5. However, because T-1 is sorted in order of increasing frequency and T-5 is sorted by code beginning with the largest frequency (and therefore the shortest code), it is necessary to invert the T-1 listing before merging. Upon merging, the code from T-5 replaces the frequency in the third computer word of T-1. The inverted tape T-1 with the substituted codes becomes tape T-7.

The transmitter dictionary tape T-8 is obtained from tape T-7. All non-alphabetic symbols are removed from T-7 and are separately listed for the punctuation and numeral tables required by the enciphering program. The suffixes, designated by a distinctive SYNTAG, are also removed from the T-7 tape and are separately listed in a suffix table used in the decomposition routine of the deciphering program.

The remaining entries on T-7 are the dictionary words. These are alphabetically sorted and the resulting tape is the T-8 tape required at the transmitter terminal.

Standard conversion and format routines produce output copy from the T-7 and T-8 tape data.

Statistical and Linguistic Analysis Programs (Chart 6)

The adaptive feature of the information transmission system is encompassed in a number of computer programs which employ standard data processing procedures. Chart 6 shows the basic programs and the tapes produced by each program.

Tape T-9 is produced at the termination of a message processing period by the deciphering program. Essentially it is a dump of the frequency counters of all dictionary words with non-zero entries. In addition, counts of individual letter, symbol, and suffix frequencies are dumped on T-9. A count of the total number of input characters, total number of words processed, and their division into dictionary, synthesized, and spelled categories is also made. Tape T-11 is produced by sorting tape T-9 on frequency.

T-10 is the new word tape on which appears every word that is either synthesized or spelled out. The T-10 tape is processed by dividing the words into their separate categories and cumulating word counts. After sorting each category according to frequency, tapes T-12 and T-13 are produced.

In actual practice, T-9 and T-10 are included on one tape and T-11, T-12, and T-13 can be considered as intermediate tapes used to produce T-15. Tape T-15 is the adaptive word list consisting of a merge of the dictionary, synthesized, and spelled words, ordered on frequency. Tape T-14 is an expanded version of T-15 and is used for producing output copy in a desired format.

T-15 is merged with the original T-1 tape when a new encoding of the dictionary is appropriate.

Computation Programs

Three programs are used to determine the efficiency of the encoding procedures and the average length of a dictionary word.

One program calculates the information content of the dictionary: $-\sum p_i \log_2 p_i$. A second program is used to compute the summation of the product of relative frequency and code length for each dictionary entry: $\sum p_i L_i$.

A third program was written to calculate the summation of the product of number of letters and the frequency of each dictionary word and the average letters per word.

B. Message Enciphering

The basic flow charts for the encipher program at the transmitter are shown in Charts 7. 1 through 7. 12. These flow charts are general and machine-independent except for reference to a drum to core transfer in the dictionary search portion. Although the description which follows leans heavily on the program developed for the UNIVAC 1105, the logic can be readily applied to another machine. Chart 7. 1 presents the basic flow of the program while the remaining charts show the various subroutines in detail. Chart 7. 13 is the output routine for the transmitter program without error checking. Charts 7. 14 through 7. 18 are substituted for Chart 7. 13 when the error detection procedures are employed.

The basic symbols used in the computer are shown in Table 24, the octal equivalent or XS-3 representation of the six-bit BCD code appearing in the first column. There are more characters available on the off-line flexowriter and the off-line high speed printer but characters were selected for compatibility with the two devices, thus insuring that input and output text would appear the same without character replacement.

Computer symbols that are compatible with both input and output equipment are divided into nine classes:

1. Alphabetic word forming characters: A - Z
2. Numeric and special characters: 0 - 9 $ / * +
3. Word terminators other than space: , ; :) -
4. Space

XS-3	SYMBOL	XS-3	SYMBOL
00	NOT USED	40	NOT USED
01	SPACE	41	NOT USED
02	- DASH, MINUS	42	NOT USED
03	0	43	) RIGHT PARENTHESIS
04	1	44	J
05	2	45	K
06	3	46	L
07	4	47	M
10	5	50	N
11	6	51	O
12	7	52	P
13	8	53	Q
14	9	54	R
15	' APOSTROPHE	55	$ DOLLAR SIGN
16	NOT USED	56	* ASTERISK
17	(LEFT PARENTHESIS	57	NOT USED
20	NOT USED	60	STOP - EOM
21	, COMMA	61	NOT USED
22	. PERIOD	62	: COLON
23	; SEMI-COLON	63	+ PLUS
24	A	64	/ SLASH
25	B	65	S
26	C	66	T
27	D	67	U
30	E	70	V
31	F	71	W
32	G	72	X
33	H	73	Y
34	I	74	Z
35	NOT USED	75	NOT USED
36	NOT USED	76	NOT USED
37	NOT USED	77	NOT USED

TABLE 24. BASIC COMPUTER SYMBOLS

5. Period or decimal point: .

6. Left parenthesis: (

7. Apostrophe, used for both left and right quotation marks: '

8. End of message: stop 60_8

9. Not Used: Error printout if used in input text.

Possessives and contractions were excluded from input text because the apostrophe was preempted to serve as quotation marks. Similarly, certain symbols such as ? ! ¢, etc., were also excluded because they were not compatible with the computer and both input and output devices.

The Storage Map for the UNIVAC 1105 program is shown in Figure 29. This is for an equipment configuration of the UNIVAC 1105 of two magnetic core storage units (8, 192 computer words of 36 bits each) and a magnetic drum of 16, 384 computer words.

The first region in the Storage Map of Figure 29 provides storage for all dictionary words starting with A, B or C. Since the dictionary format consists of four computer words, as shown in Chart 1, a maximum of 1, 128 dictionary words can be stored in core storage. The remaining dictionary words are stored on drum. As discussed in Section VII, a reorganization of the three-letter table in the same storage region can significantly reduce processing time if words stored in core-storage are selected on the basis of the most frequently used first letter.

The next region provides storage for the suffix table. A maximum storage for 64 suffixes is provided although only 43 suffixes are used in the experimental dictionary. The format of the suffix table is similar to the dictionary format, four computer words being provided for each suffix.

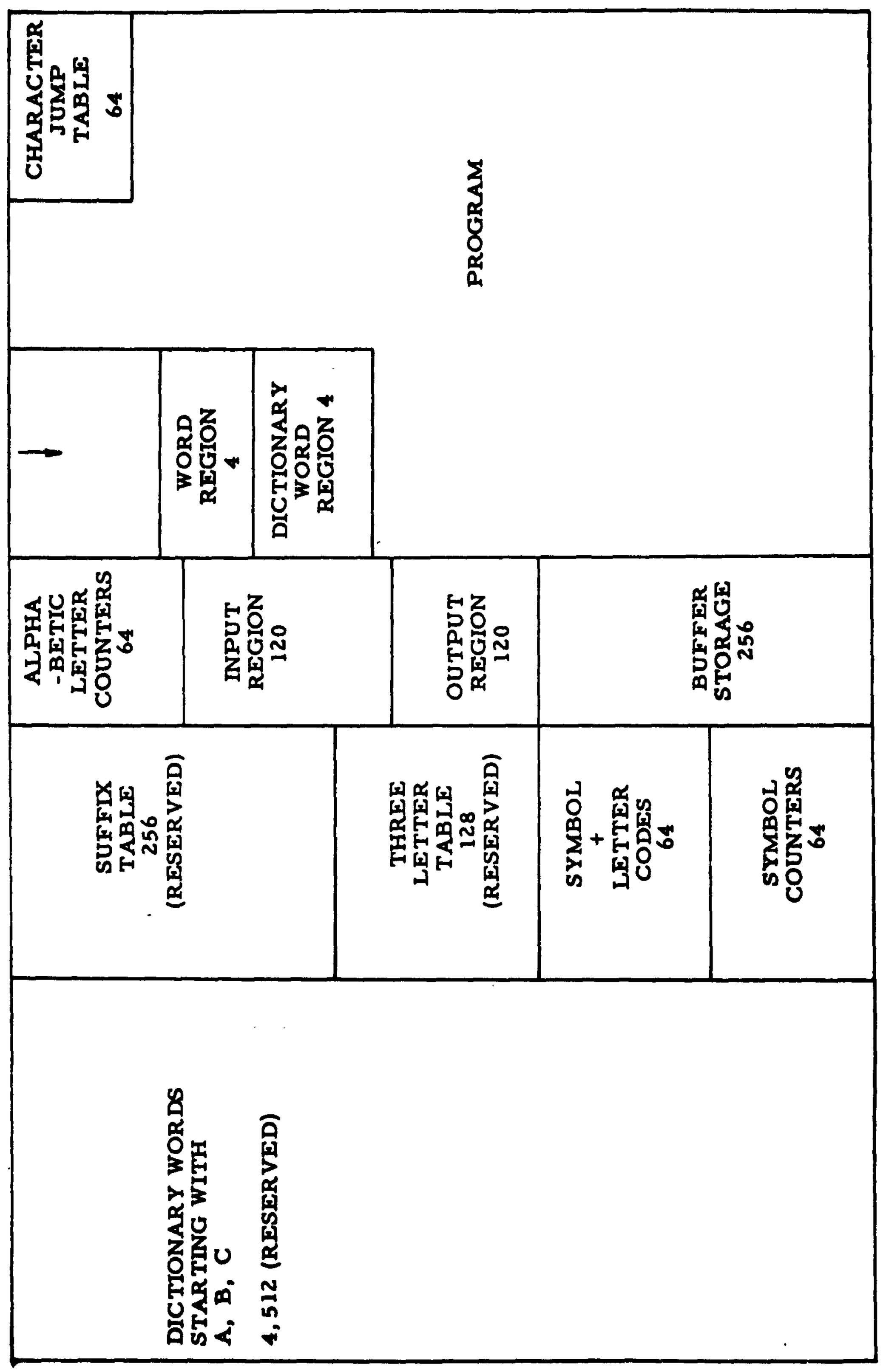


FIGURE 29. STORAGE MAP OF ENCIPHER PROGRAM

The three-letter table follows the suffix table; its use in search and decomposition will be discussed later. The format of the three-letter table is shown below:

0	$\ell 1$	$\ell 2$	$\ell 3$	ADDRESS
3 BITS	6	6	6	15 BITS

Each entry consists of one word with three zero bits preceeding the three letters. The address is the last storage location for the words starting with the designated three letters. Thus the first three letters of any input word whose numerical value is less than the three letters in a given position of the table, and greater than the three letters in the preceeding entry, will be found within the address range of the two words. A maximum of 128 three letter entries is provided in the table, 93 being used for the experimental dictionary. For the UNIVAC 1105 program, an address from the three-letter table which is greater than 37777_8 causes a drum to core transfer of these words to the buffer storage that is provided. All addresses 40000_8 or over refer to information stored on drum

The next region consists of 64 computer words for storage of the binary code for all special symbols and letters shown in Table 24. The relative position of these codes within the region is also according to Table 24. The codes for A-Z are based on an encoding for a 27 character alphabet and are used in letter mode transmission. The space code of this encoding is used as the code for exit from the letter mode.

Three special codes are also stored in this region. Their relative position (octal) and use are:

00 Exit letter mode

01 Space code (used to remove ambiguity in punctuation)

77 Enter letter mode.

Since spaces are not normally transmitted between words, the space code is necessary to provide proper spacing between numeric and special characters.

The next two regions provide storage for symbol and letter counters. The first region of 64 locations is used for all symbols and special use of alphabetic characters. The second region of 64 counters is used for letter counters. The distinction between the two regions is that the second gives the frequency of use of letters in English words while the symbol counts in region one give the frequency of use of letters in mixed alphanumeric words such as M22, U235, or 1040A. Since only 26 positions would be used in region two, other counters are stored in this region. The relative position (octal) of these special counters in the Alphabetic Letter Counter region and their use are:

01 - total number of characters processed.

02 - total number of word tokens.

03 - total number of words found in dictionary.

04 - total number of words synthesized from a stem and suffix.

05 - total number of bits transmitted.

06 - total number of entries into letter mode.

07 - total number of spaces in input text.

10 - total number of decimal points used.

11 - total number of word types located in the dictionary.

The next two regions of 120 computer words each are for input and output, their size being determined by the input-output buffer of the UNIVAC

1105.

The Buffer Storage region is used as a working storage region for words transferred from drum to core. A maximum of 64 dictionary words can be contained in this region. Thus any transfer from drum to core will be equal to or less than 256 computer words.

The last two regions which are important in understanding the program is the Word Region and Dictionary Word Region. The Word Region is used to form the token or input word from the input message. As the letters of a word are stored in the word region a Letter Count is made in word 4 of the Word Region. A word token may contain up to 18 letters.

The Dictionary Word Region is a working storage region which is used as a temporary storage for a dictionary word. Operations such as determining the longest match and stripping suffixes are performed in this region.

The remainder of Storage is devoted to the program. Only the Character Jump Table which is internal to the program is of interest. This table consists of 64 unconditional transfer instructions which direct the flow of control of the program to one of nine subprograms corresponding to the nine classes of characters defined previously. In this manner, a reinterpretation of any character may be made by simply changing the transfer instruction in the Character Jump Table and adding a subprogram which corresponds to the new interpretation.

Chart 7. 1, GENERAL INTERPRETATION, shows the basic program for the nine classes of characters. The first portion of the program consists of initialization and general interpretation. There are two initialization procedures. Initialization 1 is for system initialization. After the entire system tape is read into core and drum, all word, suffix, and symbol counters are set to zero. Initialization 2 is used to initialize between messages. None of the counters are reset except for the Letter Count. During Initialization 2 the Input Region and Word Region are filled with spaces and the Output Region and Letter Count are filled with zeros.

Selection of the input media, magnetic tape or paper tape, is performed prior to START by setting a console switch. The PAPER TAPE READ and MAGNETIC TAPE READ are standard subroutines. Reading is performed in blocks.

Each character is picked up from the input string and saved in a register called CHAR. A count is kept of the total number of characters processed. The character is then added to the base address of the Character Jump Table which has been discussed previously. An unconditional jump is made to the Table with this address and the jump command stored in the Table directs control to one of the nine subprograms which follow.

Class α_1 includes all alphabetic characters. If neither the LETTER MODE nor the SPECIAL CHARACTER switch is set the character is sent to the Word Region and the count of the character type is incremented in the Alphabetic Counter Region. The LETTER COUNT of the word being accumulated in the Word Region is also incremented and tested to see if it is equal to 18. If the LETTER COUNT is equal to 18, then the previous letters accumulated in the Word Region are sent in letter mode by the TRANSMIT LETTERS IN WORD subroutine. This subroutine also sets the LETTER MODE switch and control returns to pick up the Next Character (NC). If the next character is alphabetic, then the positive branch of the LETTER MODE switch is taken. This causes the next alphabetic character to be sent in letter mode by the INST subroutine, Chart 7. 3, which transmits the code for this character and increments the counter in the Symbol Region. Thus if a word is greater than 18 letters in length, the character count is kept in the Symbol Region as a special use of alphabetic characters. If a special character has been the last character encountered, the SPECIAL CHARACTER switch will have been set and the alphabetic character will be sent in letter mode via INST after the ENTER LETTER MODE subroutine.

Class α_2 includes all the special characters. The program consists of an entry to the SPCHR subroutine, which is shown in Chart 7. 2, and an entry to INST through connector CS. The essential part of the program is the SPCHR subroutine which will be described here. Referring to Chart 7. 2, the first action to be taken is the setting of the SPECIAL CHARACTER SWITCH. Since all special characters are sent in word mode, the LETTER MODE switch must be tested. If it is set, it must be reset through the EXIT LETTER MODE subroutine. The LETTER COUNT is tested next to see if a word has been accumulated in the Word Region. If the LETTER COUNT is greater than zero then the letters accumulated are sent in letter mode by the TRANSMIT LETTERS IN WORD subroutine, Chart 7. 10. Before the letters are transmitted the present special character must be saved and then restored before exit from the subroutine.

Class α_3 includes most word terminators other than space. The basic part of this program is the TERM subroutine which is shown in Chart 7. 4. The TERM subroutine first tests the LETTER MODE switch to determine what mode the system is in. Since all terminator symbols are transmitted in word mode, if the letter mode switch is set, then an entry is made to the EXIT LETTER MODE subroutine before continuing. The LETTER

COUNT is tested next to see if a word has been accumulated in the Word Region. If the system had been in letter mode prior to encountering a terminator symbol then the LETTER COUNT would be zero and the DICTS subroutine would be by-passed, control going directly to EXIT. This test of the LETTER COUNT also provides for the case of a period used as a decimal point and double punctuation. If the LETTER COUNT is not equal to zero, then entry is made to the DICTS subroutine where search and decomposition is performed. The DICTS subroutine is the most extensive subroutine and will be described later. After the DICTS subroutine the Word Region and the ALPHA routine or α_1 program is initialized (connector X_1). The exit for subprogram α_3 is through connector CSR which is in the class α_4 program, which is described next.

Class α_4, the space routine, begins with an entry to the TERM subroutine which has been described previously. Normally no space is transmitted between words; only if a special character has been encountered will it be necessary to transmit the special space code. This provides for proper spacing between numeric and alphabetic words. Thus if a space code were not provided, strings such as 49 50 and 33 MEN would appear in the output of the receiver routine as 4950 and 33MEN. The TEST SPECIAL CHARACTER SWITCH branch in the space routine provides for this spacing problem. If the SPECIAL CHARACTER SWITCH is not set, then only a count is kept of the spaces which occur in the input text by means of the special counter provided in the Alphabetic Counter Region. If the SPECIAL CHARACTER SWITCH is set, then a space is sent via INST and the SPECIAL CHARACTER SWITCH is reset. Connector CSR provides an entry for portions of the program which deal with other terminators that require resetting of the SPECIAL CHARACTER SWITCH.

The Class α_5 program distinguishes between a period and decimal point by testing the status of the SPECIAL CHARACTER SWITCH. The same binary code is transmitted for either use but a separate count is made for the period used as a decimal point. Exit is through connector CSR to reset the SPECIAL CHARACTER SWITCH.

For the left parenthesis, Class α_6, the LETTER COUNT is tested to see if any letters have preceded the symbol (. If some letters have been accumulated in the Word Region, an error type-out occurs and entry to the SPCHR subroutine is made. If the LETTER COUNT is zero then the normal use of (has been made and exit is through connector CS.

Class α_7, the apostrophe, is used as quotation marks. This symbol is handled in a special way to insure proper spacing in the receiver program. Since the same symbol, ' , is used as the left and right quotation mark a distinction is made between the two by their order of occurrence. The QUOTE SWITCH is set when the first apostrophe is encountered and reset on the next occurrence of this symbol. The first action taken is testing the status of the QUOTE SWITCH. If it is not set, then it is set in the next step and the symbol is saved. The LETTER COUNT is next tested to see if any letters have been accumulated in the Word Region. This procedure is similar to the manner in which the left parenthesis was handled. If the LETTER COUNT is greater than zero, then an ERROR typeout occurs and an entry to the TERM subroutine is made. SEND ONE SPACE provides an extra space for proper spacing in the receiver output. The character, ', is restored in CHAR and exit is through connector CS.

The other branch of this program is taken when the QUOTE SWITCH is set. The action to be taken is that for a terminator corresponding to a close quote, ". The QUOTE SWITCH is reset and the TERM subroutine is entered. The symbol is transmitted via INST and an extra space is sent for proper spacing in the receiver. Exit is through CSR to reset the SPECIAL CHARACTER SWITCH.

Class α_8 is the end of message symbol. The EOM symbol is first treated as a regular terminator through the TERM and INST subroutines. The information remaining in the output region must be transmitted, requiring that the OUTPUT subroutine be set for the last block to be transmitted. With the End of Message symbol, processing halts. The next message will restart with Initialization 2.

The last class, α_9, is provided for all unused characters that may occur in the input text. An error printout is made on the on-line Flexowriter of the character in octal. The unused character printout and other error typeouts are shown in Table 25.

The DICTS subroutine, Charts 7.5 and 7.6, which include dictionary search and word decomposition, is the longest and most complex subroutine. Entry is through the TERM subroutine, Chart 7.4. The initial step in the subroutine consists of obtaining the first three letters of the input word stored in the Word Region. A table look-up of these three letters is made in the Three-Letter Table. The Table used in the experimental system is shown in Table 26 with the number of dictionary words found in each three-letter group.

If the three letters of the input word exceed the last entry of the table, e.g., ZOO, the word is transmitted in letter mode by the TRANSMIT

00	ILLEGAL CHARACTER
01	THREE LETTER TABLE EXCEEDED
02	ERROR IN DRUM STORAGE ALLOCATION ($>$ 256)
03	WORDS IN THREE-LETTER GROUP NOT A MULTIPLE OF FOUR
04	ERROR IN CODE
15	LETTERS PRECEDE OPEN QUOTE
16	ILLEGAL CHARACTER
17	LETTERS PRECEDE LEFT PARENTHESIS
20	ILLEGAL CHARACTER
35	ILLEGAL CHARACTER
36	ILLEGAL CHARACTER
37	ILLEGAL CHARACTER
40	ILLEGAL CHARACTER
41	ILLEGAL CHARACTER
42	ILLEGAL CHARACTER
57	ILLEGAL CHARACTER
60	ILLEGAL CHARACTER
61	ILLEGAL CHARACTER
75	ILLEGAL CHARACTER
76	ILLEGAL CHARACTER
77	ILLEGAL CHARACTER

TABLE 25. ERROR TYPEOUTS
(ON-LINE FLEXOWRITER)

ADJ	63	FOR	60	RAC	58
ALL	60	FUN	61	REB	51
ANX	60	GIR	61	REJ	52
ART	62	GOW	40	REQ	42
AVO	59	GRO	56	RIC	55
BAT	64	HAT	60	RUM	60
BIG	63	HIG	62	SAY	56
BOO	60	HOV	61	SEI	56
BRI	54	IMM	60	SEX	45
BYO	60	INE	54	SHO	56
CAT	61	INS	52	SIZ	59
CHE	60	JAW	59	SOA	59
CLE	58	KNE	59	SPA	55
COL	53	LAZ	48	SQU	54
COM	48	LIL	60	STI	56
CON	94	LOU	59	STR	54
CRA	56	MAM	48	SUN	60
CUT	67	MEC	56	SWU	57
DEC	58	MIR	58	TEM	53
DER	46	MOT	55	THI	57
DIN	58	NEA	57	TOG	60
DIT	54	NOS	53	TRA	59
DRE	56	ODO	50	TWE	55
ECS	60	OUN	60	UNT	55
ENC	59	PAP	57	VER	58
EUR	55	PEE	60	WAN	56
EXI	49	PIG	58	WHA	56
FAL	59	POI	60	WIS	55
FIF	60	PRA	56	YEL	60
FLI	53	PRI	48	ZON	15
		PRO	60		

TABLE 26. THREE-LETTER TABLE

WORD BY LETTERS subroutine, Chart 7. 11, and exit is at connector EE.

The format of the three-letter table has been discussed previously. The information obtained from the table is the address range in which a dictionary word which is similar to an input word can be stored. The first address is tested to see if it is a drum address. If it is, the words are transferred from the drum to the Buffer Storage Region in core.

The search for the longest match employs the bracketing method or binary search. In this method, the dictionary of N words is ordered alphabetically in the memory. The search begins by comparing the input word with the dictionary word at the N/2-th word. If the input word is less than the dictionary word then the interval between the first word and the word at N/2 is divided into two. If the input word is greater than the word at N/4 then the interval between N/4 and N/2 is divided into two. This process is continued until the word is identified or the search fails. A total of $\log_2 N$ comparisons is the maximum number of comparisons that must be made in this method of word search.

The computer program begins the binary search at X_1. The program is entered from either branch of the drum address test. The address range for the search is set, FWA and LWA specifying the lower and upper limits, respectively, of the search region. The number of comparisons that will be required is determined by the number of words in the respective three-letter group.

The binary search either finds an exact match of an input word with a dictionary word or terminates before the next largest dictionary word. A successful match leads to transmission of the associated code by the TRANSMIT CODE subroutine and incrementing the word counter.

The search for a longest match starts at DCM in Chart 7.5 and searches the region which contains the first three letters of the input word. The search is bidirectional, starting at A_i, the address of the dictionary word of next largest value. The BF switch is a backward-forward switch designating the direction of search. The K switch ensures that the region has been searched in both directions before the search is terminated at TWD and the input word is transmitted in letter mode.

A_j is replaced by A_i, the BF and K switches are set to 1, and a test is made at C_6 to determine that the first three letters of the input word correspond to the first three letters of the dictionary word. This test restricts the search to only that portion of the three-letter region in which the input word can be found. In Table 26, for example, it is noted that 59 words appear in the region SIZ to SOA. From Table 7, Initial Trigrams-First Letter S, it is noted that 18 sets of trigrams appear in this region, ranging in size from one word starting with SIZ to nine words beginning with SKI. The three-letter test will, therefore, restrict the word search to subregions of from one to nine words, in this particular case.

If the initial three letters match, the program jumps to C_2 in Chart 7.6 and a test is made to determine whether the number of letters in the input text word are greater than the number of the dictionary word stored at address A_j. A positive decision leads to the word decomposition subroutine whereas a negative decision leads to a continuation of the word search. It is obvious that a dictionary word cannot be a stem of an input word if the dictionary word contains more letters.

The state of the BF switch at C_3 in Chart 7.6 determines whether a continuing search is backward, C_8, or forward, C_7. In the former case, the address A_j is decremented; in the latter case, A_i is incremented.

If both the initial three-letter and number comparisons are positive, the word decomposition program is entered. This program partitions a word into stem and suffix while adhering to the grammatical rules laid down for the four major SYNTAG classes. At RS, the longest match, that is, the longest stem in a dictionary which is contained in an input word, is determined.

The determination of a longest match directs the program to the word stripping routine at STR in Chart 7. 6. The longest match is subtracted from the input word and the difference is tested to determine whether it is a listed suffix. If a listed suffix is found, the codes for both stem (dictionary word) and suffix are obtained and transmitted via TRANSMIT CODE and connector X_2. The SYNTAG 2 rule, doubling of the final consonant, is checked in this portion of the program.

Failure to find a longest match directs the program to the SYNTAG 1, delete final E, and SYNTAG 3, change final Y to I, procedures.

If a longest match is not found at A_j after word stripping and SYNTAG changes, the program returns to the bidirectional search routine at C_3.

Examples of the word search and decomposition procedures appear in Figures 30 - 35 for each of the SYNTAG classes. Numbers in parenthesis designate the number of letters in a word. The address locations are arbitrary. The arrow in each figure points to the location where the binary search has terminated.

When an input word is successfully decomposed, the codes for both the stem and the suffix are transmitted and the counter for the suffix is

CONDITION: SYNTAG 0 - STEM $<$ TEXT WORD

	101	EXAMPLE	(7)
	102	EXCEED	(6)
EXCEEDS (7) ··· >	103	EXCEEDINGLY	(11)
	104	EXCELLENT	(9)
	105	EXCEPT	(6)

$A_i = 104 = A_j$

BF = 1, K = 1

EXC = EXC

$7W_T < 9W_j$

BF $\neq$ 0

$A_j = A_j - 1 = 103$

EXC = EXC

$7W_T < 11W_j$

BF $\neq$ 0

$A_j = A_j - 1 = 102$

EXC = EXC

$7W_T > 6W_j$

EXCEED $\subset$ EXCEEDS

EXCEEDS - EXCEED = S

S IN SUFFIX TABLE

TRANSMIT: EXCEED+S

FIGURE 30. WORD DECOMPOSITION-SYNTAG 0

$$\begin{tikzcd}[row sep=small]
 & \text{CONDITION: } \underline{\text{SYNTAG 1 - STEM} > \text{TEXT WORD}} \\
 & 101 \quad \text{EXCESS} \quad (6) \\
 & 102 \quad \text{EXCHANGE} \quad (8) \\
\text{EXCITATION (10)} \arrow[r] & {} \\
 & 103 \quad \text{EXCITE} \quad (6) \\
 & 104 \quad \text{EXCITEDLY} \quad (9) \\
 & 105 \quad \text{EXCITEMENT} \quad (10)
\end{tikzcd}$$

$A_i = 103 = A_j$

BF = 1, K = 1

EXC = EXC

$10W_T > 6W_j$

EXCITE $\not\subset$ EXCITATION

SYNTAG $(W_j) = 1$

EXCITE - E = EXCIT

EXCIT $\subset$ EXCITATION

EXCITATION - EXCIT = ATION

ATION IS SUFFIX

TRANSMIT: EXCITE+ATION

FIGURE 31. WORD DECOMPOSITON-SYNTAG 1.

CONDITION: <u>SYNTAG 2 - STEM $<$ TEXT WORD</u>

	101	GESTURE	(7)
GETTING (7) $\longrightarrow$	102	GET	(3)
	103	GHASTLY	(7)

$A_i = 103 = A_j$

BF = 1, K = 1

GET $\neq$ GHA

$K \neq 0$

$K \leftarrow 0$

103 = 103

$A_j = 103 - 1 = 102$

GET = GET

$7W_T > 3W_i$

GET $\subset$ GETTING

GETTING - GET = TING

SYNTAG $(W_j) = 2$

LAST LETTER OF STEM = FIRST LETTER OF SUFFIX = T

TING - T = ING

ING IS SUFFIX

TRANSMIT: GET+ING

FIGURE 32. WORD DECOMPOSITION-SYNTAG 2.

CONDITION: <u>SYNTAG 3 - STEM > TEXT WORD</u>

	101	COOPERATION	(11)
	102	COP	(3)
COPIOUS (7) $\longrightarrow$	103	COPE	(4)
	104	COPPER	(6)
	105	COPY	(4)
	106	CORAL	(5)

$A_i = 104 = A_j$

K = 1, BF = 1

COP = COP

$7W_T > 6W_j$

COPPER $\not\subset$ COPIOUS

$A_j = 104\text{-}1 = 103$

$7W_T > 4W_j$

COPE $\not\subset$ COPIOUS

SYNTAG $(A_j) = 1$

COPE - E = COP

COP $\subset$ COPIOUS

COPIOUS - COP = IOUS

<u>IOUS</u> $\neq$ SUFFIX

SYNTAG $(R_j) \neq 2$

$A_j = 103\text{-}1 = 102$

COP = COP

$7W_T > 3W_j$

COP $\subset$ COPIOUS

COPIOUS - COP = IOUS

<u>IOUS</u> $\neq$ SUFFIX

SYNTAG $(R_j) = 2$

P $\neq$ I

$A_j = 102\text{-}1 = 101$

COP $\neq$ COO

K $\neq$ 0

K $\leftarrow$ 0

$A_i > A_j$

BF $\leftarrow$ 0

$A_j = A_i + 1 = 104 + 1 = 105$

$A_i = 105$

COP = COP

COPY $\not\subset$ COPIOUS

SYNTAG $(W_j) = 3$

COPY $\leftarrow$ COPI

COPY $\subset$ COPIOUS

COPIOUS - COPI = OUS

OUS IS SUFFIX

TRANSMIT: COPY+OUS

FIGURE 33. WORD DECOMPOSITION-SYNTAG 3.

incremented. The dictionary word counter is not incremented but the complex word in the Word Region is marked as a synthesized word and placed on the New Word tape by the NEW WORD subroutine before exit through connector EE. A word which is not in the dictionary or cannot be decomposed is transmitted by the TRANSMIT WORD BY LETTERS subroutine through connector TWD.

The remaining charts, 7. 7 to 7. 12, have been referred to in the previous discussion. There is a distinction between Chart 7. 10 and Chart 7. 11 which should be noted. Chart 7. 10, TRANSMIT LETTERS IN WORD, is used for special use of alphabetic letters and the counter in the Alphabetic Counter Region is decremented in this subroutine. The frequency count is kept in the Symbol Counter Region by INST when this subroutine is used. Chart 7. 11, TRANSMIT WORD BY LETTERS, is entered only through the DICTS subroutine and is used for words not in the dictionary. These words are also entered on the New Word tape by the WRITE NEW WORD subroutine together with information as to the total number of bits used to transmit the word in letter mode.

Chart 7. 13 is the normal OUTPUT subroutine and is used when error checking is not employed. The output of the enciphered message can be either on paper or magnetic tape.

When the error detection program is incorporated into the system, subroutines 7. 14 through 7. 18 are substituted for the normal output subroutine. These subroutines introduce the appropriate check digits in their proper position in a synchronizing sequence and determine the synchronizing sequences.

In Chart 7.14 the program checks that no computer or programming error generated an illegal code (greater than 17 digits), stores the code and the number of digits in the code and then enters the EC OUTPUT subroutine of Chart 7.15.

The EC OUTPUT subroutine increments the digit counter and tallies the one and zero digits as they are processed, one at a time, into the synchronizing sequence program beginning at X_1. The switch is set to β_2, the code breakpoint table is initilaized for the B register, a code marker is placed in BREG, and return is made to G_1. The next digit is processed through the zero and one counters and the β_2 path is followed to determine the synchronizing sequence. Since the first digit of the enciphered word was not transferred to the B register, BREG starts in state F (minus one digit) and deciphering takes place in the B register as if it were the receiver.

When the stored code length at G_1 is zero, the stored code has been processed and, following the EC STORE SEQUENCE at H_1, the B register is checked for the presence of the code marker alone. BREG = 1 indicates that a synchronizing sequence has been formed and the check digits are inserted before the sequence is transmitted. A sync frame code is stored in the first U positions of the sequence (5 digits in the experimental system), the CHECKWORD is formed (6 digits), the first V positions (4 digits) of which follow the SFC, and the zero and one parity checks follow the V digits.

The EC WRITE SEQUENCE, Chart 7.17, prepares the output for transmission via either paper or magnetic tape.

If an end of message (EOM) code is encountered, and self-synchronization has not been achieved, synchronization is forced by the method outlined in Section VIII. The appropriate C-terminator is located in the SYNC TABLE, and the EC OUTPUT subroutine is entered for output preparation of the C-terminator.

C. Message Deciphering and Decoding

The decipher and decode program at the receiver is made up of the main program and three subroutines as shown in Charts 8. 1 through 8. 4. Boxes with broken lines indicate those portions of the program that utilize the error detection procedures described in Section VIII. The elimination of these boxes converts the program to the basic decipher and decode program without error detection.

The program deciphers the received string of binary digits to form codes and identifies the word associated with each code. At various places in the deciphering and decoding program, error checks are made. As each word is decoded, it is placed in word storage and is designated "previous word". The previous word is held in word storage until the next succeeding word is decoded. The SYNTAG of both words determines what editing, if any, is required before the previous word is printed out. Spacing is provided by the SPACE subroutine.

Chart 8. 1 Decipher and Decode Message

The program starts with the reading and storing of the code break-point table, tape T-6. The storage location of the first dictionary word corresponding to each B-set (set of codes of equal length) is then determined. A dictionary word can be a word, punctuation mark, special character, or a program operator (end of message, check word, letter mode). The dictionary is read from tape T-7 and stored.

The DECODE subroutine of Chart 8. 2 checks the incoming string of binary digits for errors, isolates individual codes, and then decodes by locating the dictionary words associated with the code.

Each dictionary word is identified by a SYNTAG as previously shown in Chart 1. The SYNTAG identifies the word as a suffix, special character, terminator, period, comma, bracket, space, letter mode, operator, or end of message. The SYNTAG identification is shown in the following table.

SYNTAG IDENTIFIERS

SYNTAG	DESCRIPTION
0	Regular stem
1	Final E deleted if suffix begins with a vowel
2	Final consonant doubled if suffix begins with a vowel
3	Final Y changed to I if suffix begins with a letter other than I
4	Suffix begins with consonant
5	Suffix begins with A, E, O, U, Y
6	Suffix begins with I
7	Terminators : ;)
8	Comma ,
9	Special characters $ - / * 0-9 +
10	Brackets ('
11	Space
12	Period .
13	Letter mode operator %
14	End of message #
15	Sync frame code

If the decoded word is a stem, the WRITE PREVIOUS WORD subroutine of Chart 8.3 is called up followed by the SPACE subroutine of Chart 8.4.

If the word is a suffix, the SYNTAG of the previous word and of the current word are examined to determine the conditions for synthesis of the word. If the final letter of a stem is dropped or changed, the letter count of the previous word is reduced by one and the previous word is transferred to the output. If the final letter is changed, the proper letter is transferred to the output. If the final letter is repeated, the previous word is transferred to the output and the final letter is repeated again in the output. If no editing is required the previous word is simply transferred to the output.

If the word is a terminator, comma, period, or space, the previous word is transferred to the output.

If the word is a special character or a bracket, the previous word is transferred to the output. The SYNTAG of the previous word determines what editing is required. Spaces, as required, are transferred to the output area.

When the DECODE subroutine identifies an end of message code, the previous word is transferred to output. Under the error detection option, the sequence is also checked to determine that the message ended in synchronization. Since the identification of the EOM code has restored the A register to zero, a zero B register signifies synchronization has been achieved. The message is printed, the storage location of the previous word is cleared, and the computer operator is asked if there are more messages. If there are more messages the process continues, otherwise, the run is terminated.

If the B register is not zero when the EOM code is deciphered, the next code is deciphered. If the B register is still non-zero, ERROR 7 is printed out.

When the DECODE subroutine identifies the letter mode operator, %, the decipher program jumps from the word code breakpoint table to the letter code breakpoint table. Deciphering continues in letter mode until a space, △ , is identified after which control is returned to the word mode.

If the word is spelled, the previous word is transferred to the output together with proper spaces. The letter breakpoint table is used to decipher the input binary digits, the next code is identified, and the corresponding letter is located. If the letter is not a space, it is transferred to the output area and the next code is identified by the letter breakpoint table. If the letter is a space, the storage location of the previous word is cleared, and control is returned to the word mode.

Deciphering the code designating a sync frame code (SFC), identified by SYNTAG 15, produces a print out of ERROR 6. This is an automatic error because the SFC cannot legally appear as part of a message sequence but only in designated positions before and after a synchronizing sequence.

Chart 8.2 Decode with Error Detection

The received message is deciphered by the DECODE subroutine. After setting the α-switch, the first error checks are made on the initial check digits of the message. It will be recalled, from Section VIII, that a message begins with a sync frame code, SFC, followed by a CHECKWORD. The SFC is a five digit code and the CHECKWORD is made up of six digits, the first four (P_T) of which are the counter digits specifying the number of binary digits in the first synchronizing sequence (mod 16); the fifth digit is the number of zeros in the synchronizing sequence (mod 2), and the sixth digit is the number of ones in the synchronizing sequence (mod 2).

The digit string is read digit by digit. The first five digits must form the SFC or ERROR 1 is printed out. If an SFC code is found, the counter digits are stored for later checking with the number of digits found in the synchronizing sequence. The sum (mod 2) of the digits in the fifth (P_0) and sixth (P_1) positions of the CHECKWORD is determined and set equal to SUM. A test is made to test for a mutilated CHECKWORD by checking that SUM = P_T (mod 2). An ERROR 2 printout is obtained if this test fails.

ERROR 1 and ERROR 2 will call for retransmission of the message and deciphering is halted.

If no error has been detected, the message digits are read and deciphering begins. The deciphering and determination of the synchronizing sequence for error checking proceed in step, a digit being transferred first to the A register then the B register. After initializing designated storage locations with the initial entries of the code breakpoint table, a one is placed in the A register to serve as a code marker. The first digit of the received message is read and positioned immediately following the one in the A register. The string so constructed is compared to the maximum length code (L code) of the first B-set at CW_4. If the string is greater, the code is not contained in the first B-set, and additional digits must be read from the received message to complete the code. The next digit is read and positioned following the previously read bits, the code breakpoint table is advanced, and the new string is compared to the maximum code of the B-set. If the string is greater, the next digit is read and the process is repeated. If the string is not greater, a complete code has been read, and that code is within the B-set to which it was last compared.

In order to determine to which dictionary word in the dictionary set the isolated code corresponds, the decoding procedure at CW_5 is followed. The minimum code in the B-set is subtracted from the isolated code. If the difference is zero, the first dictionary word in the set is the desired word. If the difference is non-zero, the difference is added to the base address of the B-set and the location of the desired word is determined. The word is located and sent to the current word storage.

While the message digits are being deciphered in the A register, error checking by computing a CHECKWORD and determining a synchronizing sequence is carried out. Starting at X_2, the digit counter is incremented for every processed digit and a tally is made as to whether the digit is a one or a zero.

The synchronizing sequence is found by setting the switch to β_2 and initializing a storage location with the initial entries of the code break-point table. For purposes of error checking, it is necessary only to isolate codes; the associated dictionary words are not needed. After placing a one to serve as a code marker in the B register, the second digit of the message string is placed in the B register, thus initializing it at initial state F (minus one digit). Deciphering in the B register continues at CW_4 in the same manner as for the A register.

The synchronizing sequence is determined when both the A and B registers are simultaneously zero. When this condition occurs, the final error checks are made at CW_6. The total number of digits in the synchronizing sequence is contained in the digit counter and this number (mod 16) must equal P_T of the CHECKWORD or ERROR 3 is printed out. Similarly, the one tally must equal P_0 of the CHECKWORD or ERROR 4 is printed; P_1 must equal the one tally or ERROR 5 is printed.

Resetting the α and β switches completes the deciphering and decoding for one synchronizing sequence. The second synchronizing sequence must be preceded by an SFC code and its CHECKWORD. The SFC will at the same time serve as the terminal SFC of the first synchronizing sequence. If a synchronizing sequence is found which is not followed by an SFC, ERROR 1 is printed. Hence, the complete error detection cycle is not completed until the terminal SFC is checked.

The above procedure is repeated for all sequences through the one terminating in an EOM code, except that a terminal SFC is not required.

Chart 8.3 Write Previous Word

The "previous word" stored in the current word storage is transferred to the output area character by character. The number of characters in the word determines how many characters are transferred. The output routine is simplified inasmuch as the number of characters in a word is contained in a computer word associated with the dictionary word on tape T-7.

Chart 8.4 Space

Spaces to be inserted in the receiver output are determined by the SPACE subroutine in accordance with the space matrix. The elements in the matrix designate the action to be taken by the subroutine. Blank elements require no action. Row entries designate "previous word" and column entries designate the next succeeding word in word storage.

D. Pre-sort/Post-sort Magnetic Tape System

The Flow Chart for a system that provides for storage of the dictionary on magnetic tape is shown in Chart 9. This system has not been programmed but is considered as a representative of an alternate system

SPACE MATRIX

	WORD & LETTER MODE	SUFFIX	TERM	SPCHR	,	.	△	('	EOM
WORD & LETTER MODE	1	2	2	1	2	2	2	2	4
SUFFIX	1		2	1	2	2	2	2	4
TERM	1		2	1	2	2	2	2	4
SPCHR	2		2	2	2	2	2	2	4
,	1		2	2	2	2	2	2	4
.	3		2	2	2	2	2	2	4
△	2		2	2	2	2	2	2	4
('	2			2		2	2	2	4

$a_{ij} = 1$:	add one space
$a_{ij} = 2$:	no space
$a_{ij} = 3$:	add two spaces
$a_{ij} = 4$:	complete processing

Terminators:	, : ;)
Special Characters:	$ + - / * 0-9
% X· X· · · X △	treated as stem
△ (word mode):	treated as stem

organization to the one that has been programmed. The flow chart represents a system with an integrated dictionary alphabetically ordered on magnetic tape. The system can be modified for a split dictionary by incorporating the decomposition routine of the present system and modifying the format of the dictionary stored on tape in accordance with the three-letter grouping previously shown.

It is assumed in this system that the dictionary is stored in fixed word records of M words each on magnetic tape.

The Storage Map for the magnetic tape system is shown in Figure 34. The token or T region is for storage of the input words or symbols. The format of each word would be three computer words for the token word or symbol and one word for the letter count and precedence number. The precedence number represents the position of the word or symbol in the input message. The Precedence Region or P region would consist of two computer words, one word for the precedence number and one word for the binary code which would be found after dictionary search. The Dictionary Region would provide for storage of M words read from the dictionary tape. The next two regions would provide for input and output.

To send a word in letter mode, storage would be required for 28 letter and symbol codes: one location for the entry to letter mode, 26 positions for the alphabetic letter codes, and one for the exit code from letter mode. The remaining storage would be allocated to the program.

The first operation of the program is that of forming the token words or symbols from the input string. For each item stored in the T region the precedence number i is incremented and stored with the word token. K is initialized to zero. When i = N, or N items are stored in the T

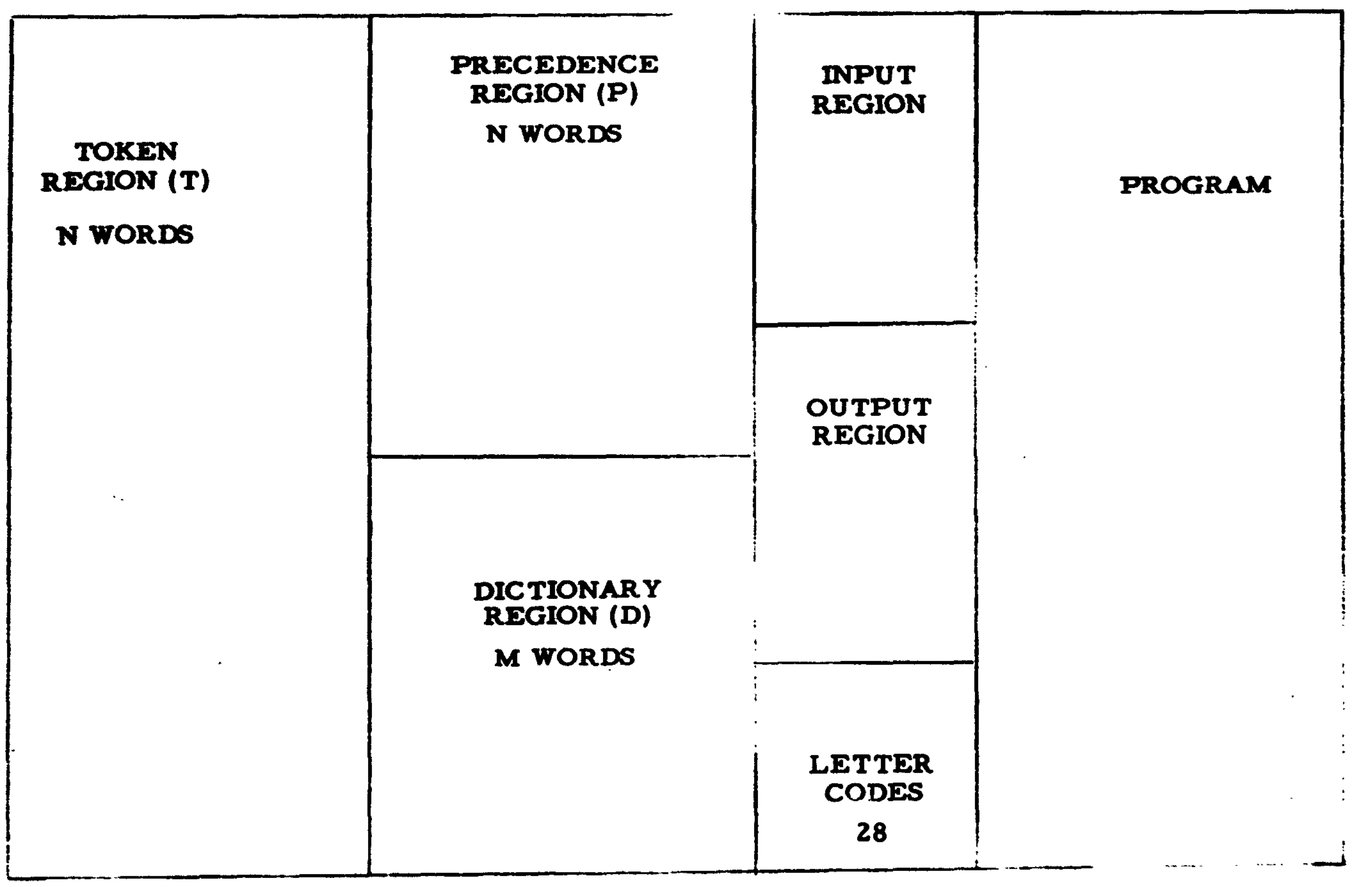


FIGURE 34. STORAGE MAP OF MAGNETIC TAPE SYSTEM

region, an alphabetic sort is performed (connector X_3). The index i is initialised at 1 and represents a running index of the items in the token region, no longer referring to the precedence number which is stored with each item. M dictionary words are then read from magnetic tape S. If an end of file is encountered, special action for words remaining in the token region are provided through connector X_{17} which is discussed later. J is a running index for the dictionary words in the D region. The token word i is first compared with the last dictionary word in the D region. If T(i) is greater than D(M) then all other words in T(i) will be greater and the next group of dictionary words must be read in. Before the next group of dictionary words are read in, the present group are duplicated on a new dictionary tape P through connector X_6.

If the last word in the new dictionary group is greater than T(i), the sequential search of the dictionary words begins with D(J). If T(i) = D(J) then by connector X_7 the precedence number and binary code are stored in the precedence region. The word counter is incremented in the D region and i is tested to see if all tokens have been found. If some items remain, i is incremented and return is by X_5 for the next token to be tested.

If $T(i) > D(J)$ then the token is not in the dictionary. Connector X_9 is the entry for new words. K is incremented and T(i) is moved to position T(K) in the token region. It is noted that K is always less than or equal to i and the entire Token Region may be used for storage of New Words. The precedence number for this token and K is stored in the Precedence Region. This item in the P region is marked as a new word. Return is through X_{10} to test if all tokens have been processed. For the case $T(i) < D(J)$ (connector X_8) the number of words in the D region (J) is tested to see if all dictionary words in this group have been used. If they have not, J is incremented and return is made to test T(i) and D(J) (connector X_5). If J = M, then the present dictionary words are copied on tape P and return is made for another group of dictionary words (connector X_4).

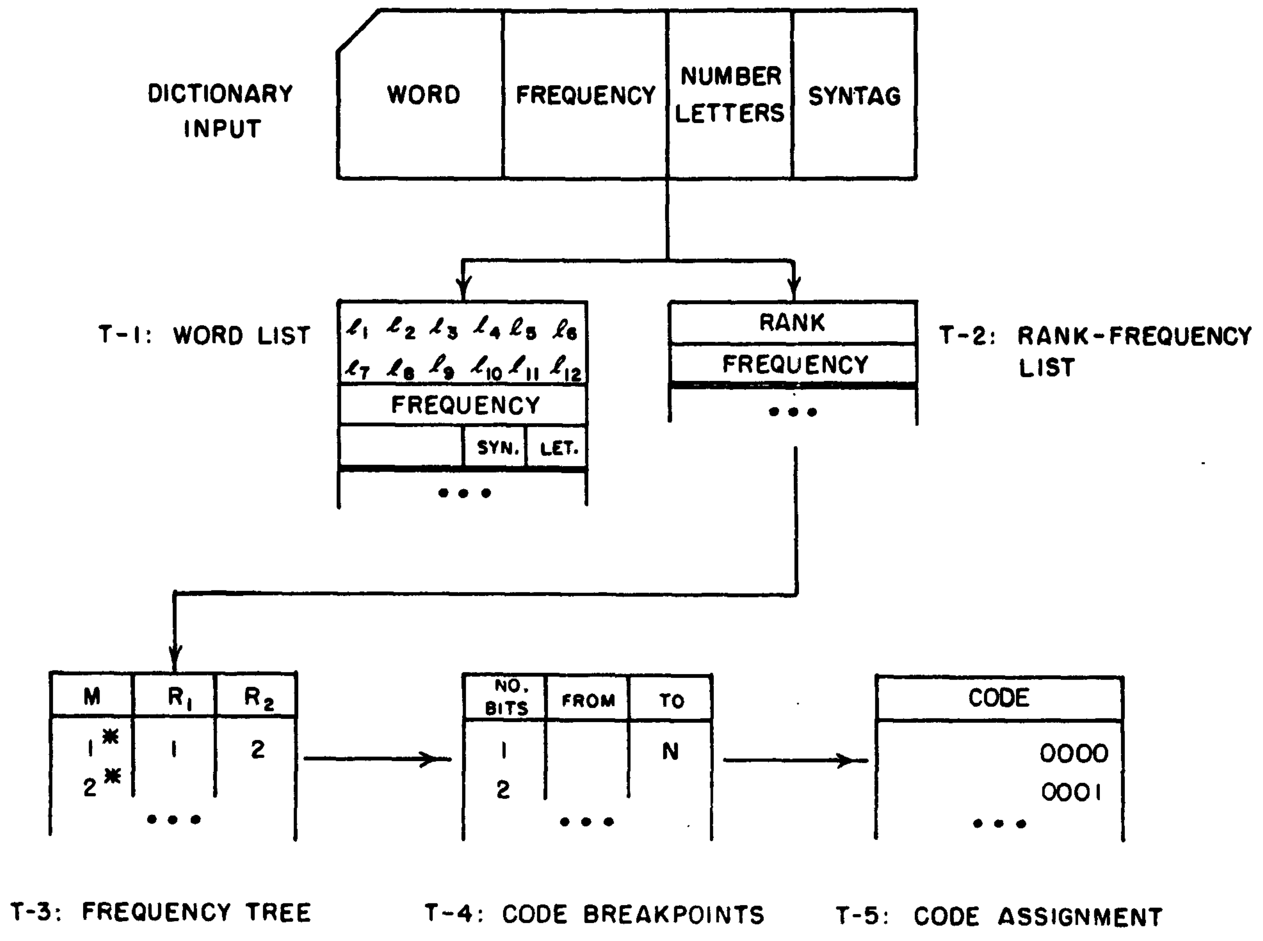
When all the items in the T region have been processed (i = N, connector X_{10}) then the P region is sorted on the precedence number and the index i is initialized at 1. Again i is used as a running index. If P(i) is a new word it is sent in letter mode (connector X_{12}). When all the items in the P region have been processed, (i = N, connector X_{14}) the K new words are written on the New Word tape and the remainder of tape S is duplicated on tape P if an end of file has not been encountered. Dictionary tape S is swapped for P and return is to X_1 for the next message. Thus two dictionary tapes are maintained with this system, one from the previous run and the updated dictionary tape with incremented word counters.

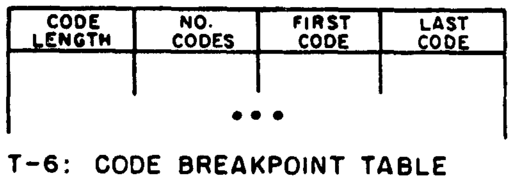
If an end of file has been encountered before all tokens have been processed, special action is taken through connector X_{17}. J is set to L, where D(L) is the item ZZ ... ZZ, greater than any token word. Return is to connector X_5 and the program continues testing and storing the items in the T region as new words since T(i) is always less than D(J) when J = L.

Timing considerations for this system organization have been discussed in Section VII. If alphabetic sorting is performed by binary or centered insertion [22] then sorting can be combined with the formation of the token word in the flow chart with a possible saving in time.

CHART I

TRANSMITTER AND RECEIVER DICTIONARY TAPES PREPARATION





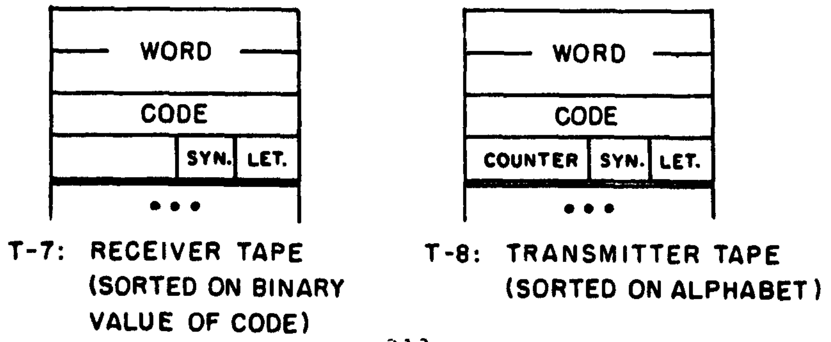

WORD
CODE
SYN.
LET.
T-7: RECEIVER TAPE (SORTED ON BINARY VALUE OF CODE)
WORD
CODE
COUNTER
SYN.
LET.
T-8: TRANSMITTER TAPE (SORTED ON ALPHABET)

CHART 2

GENERATION OF FREQUENCY TREE, T-3

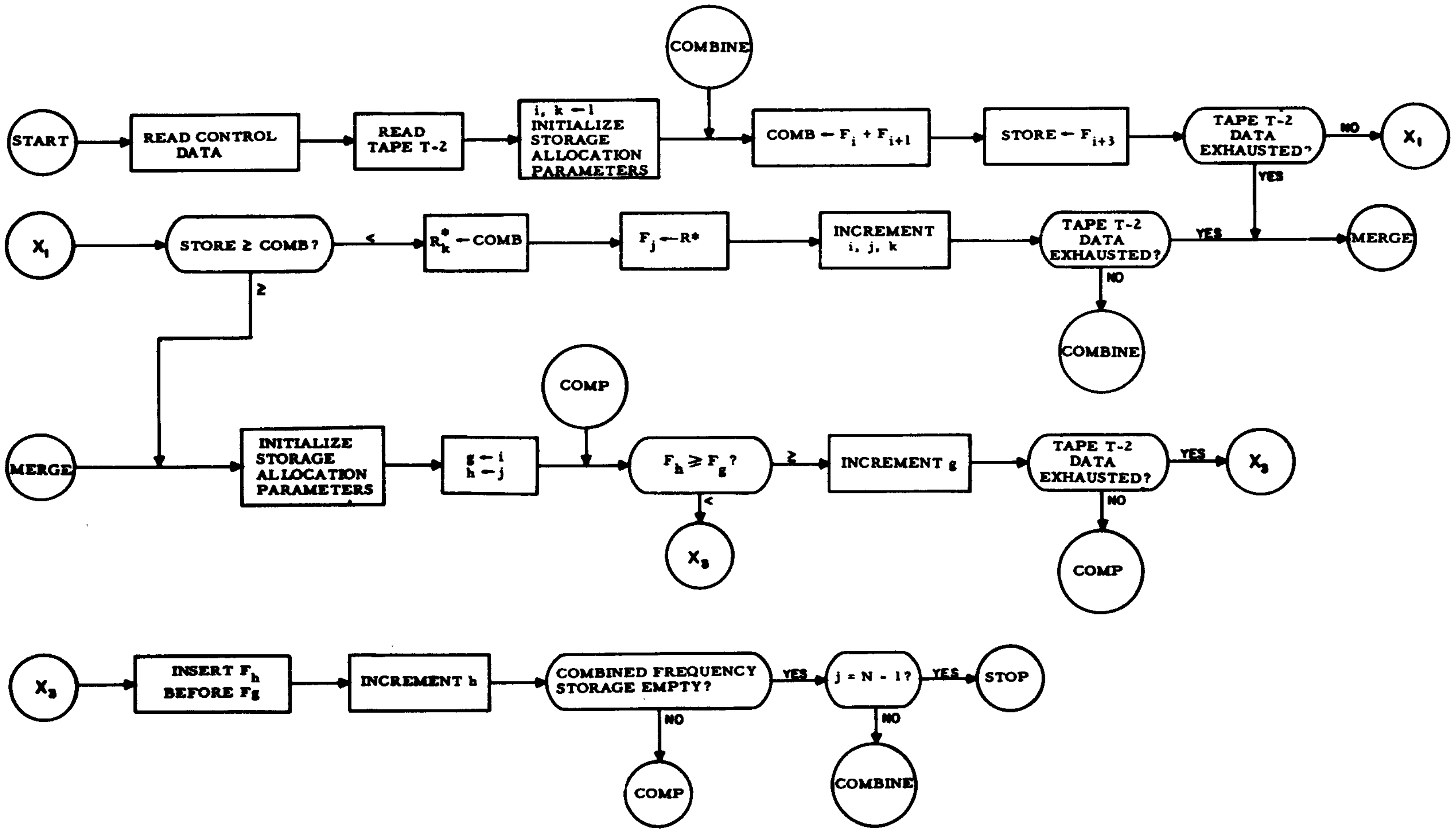


CHART 3

DETERMINATION OF CODE BREAKPOINTS, T-4

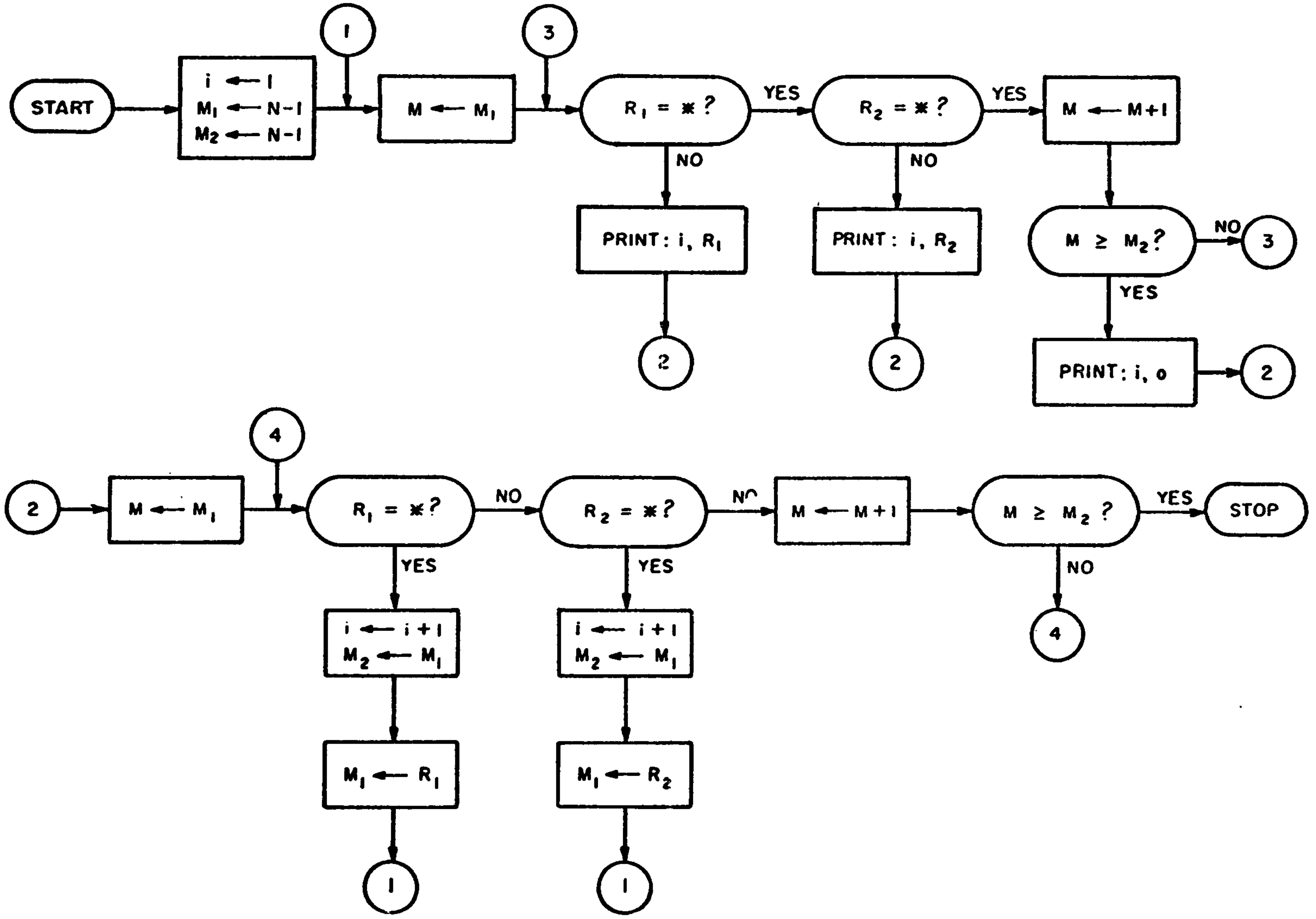


CHART 4

CODE ASSIGNMENT, T-5

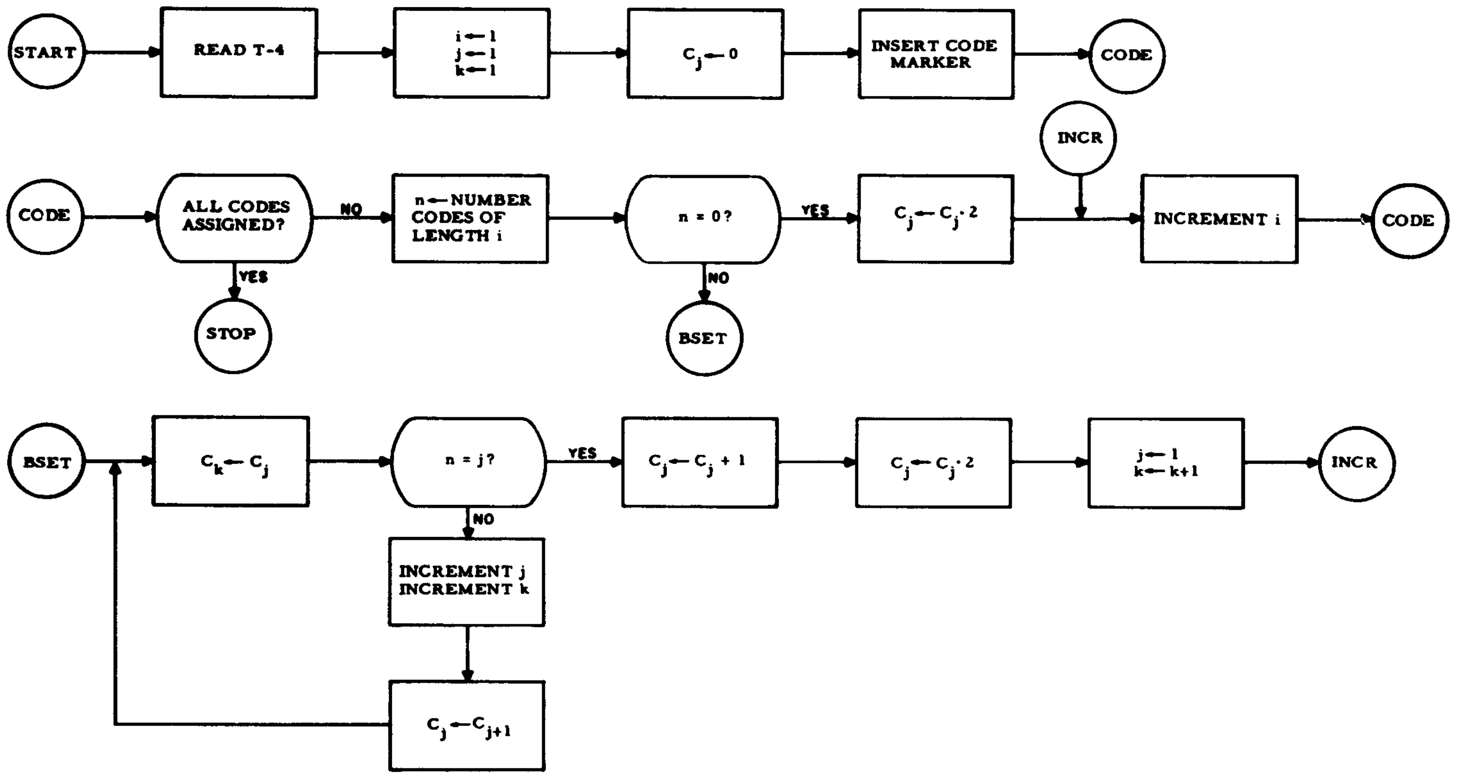


CHART 5
CODE BREAKPOINT TABLE, T-6

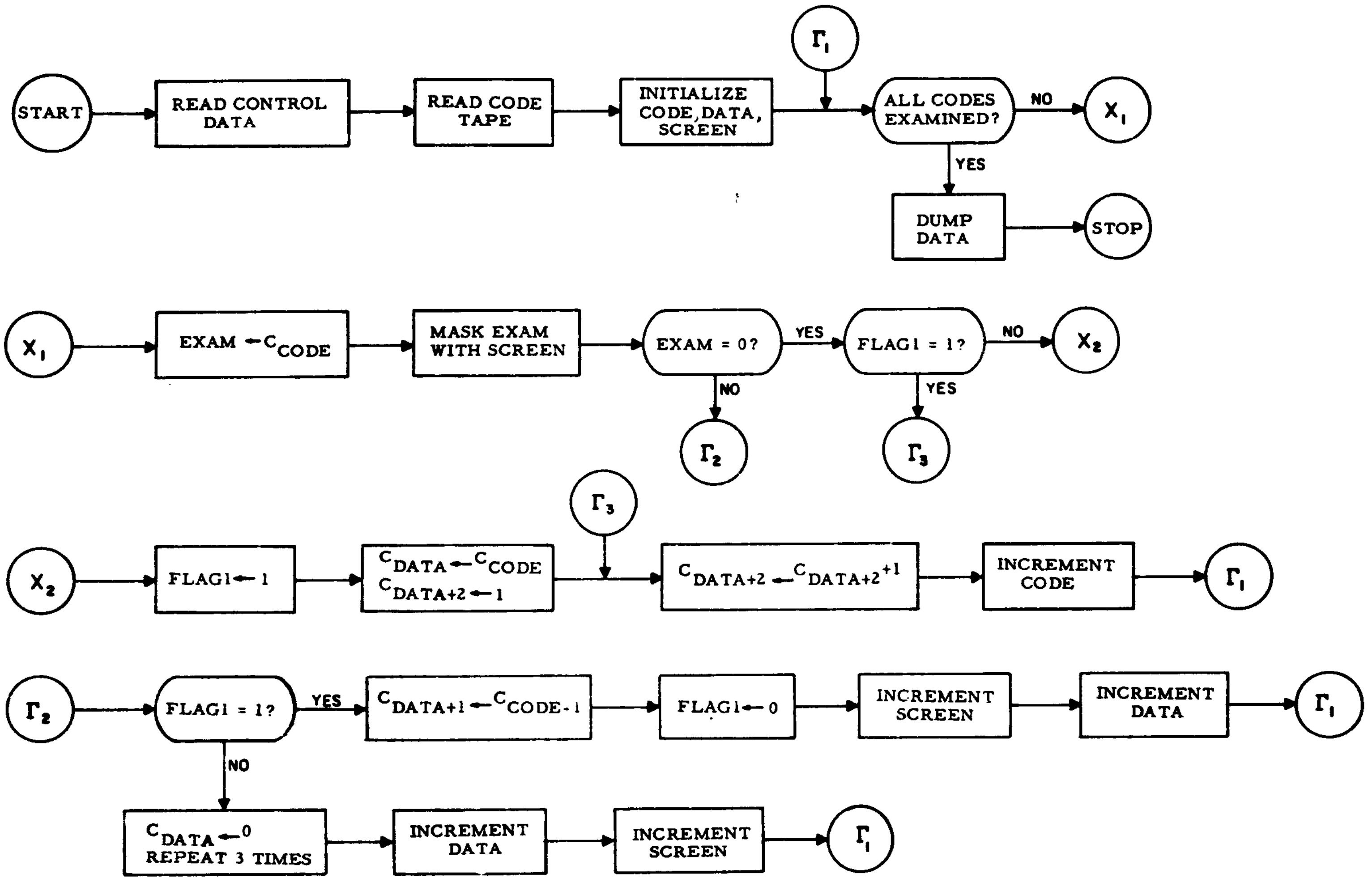


CHART 6

STATISTICAL AND LINGUISTIC ANALYSIS

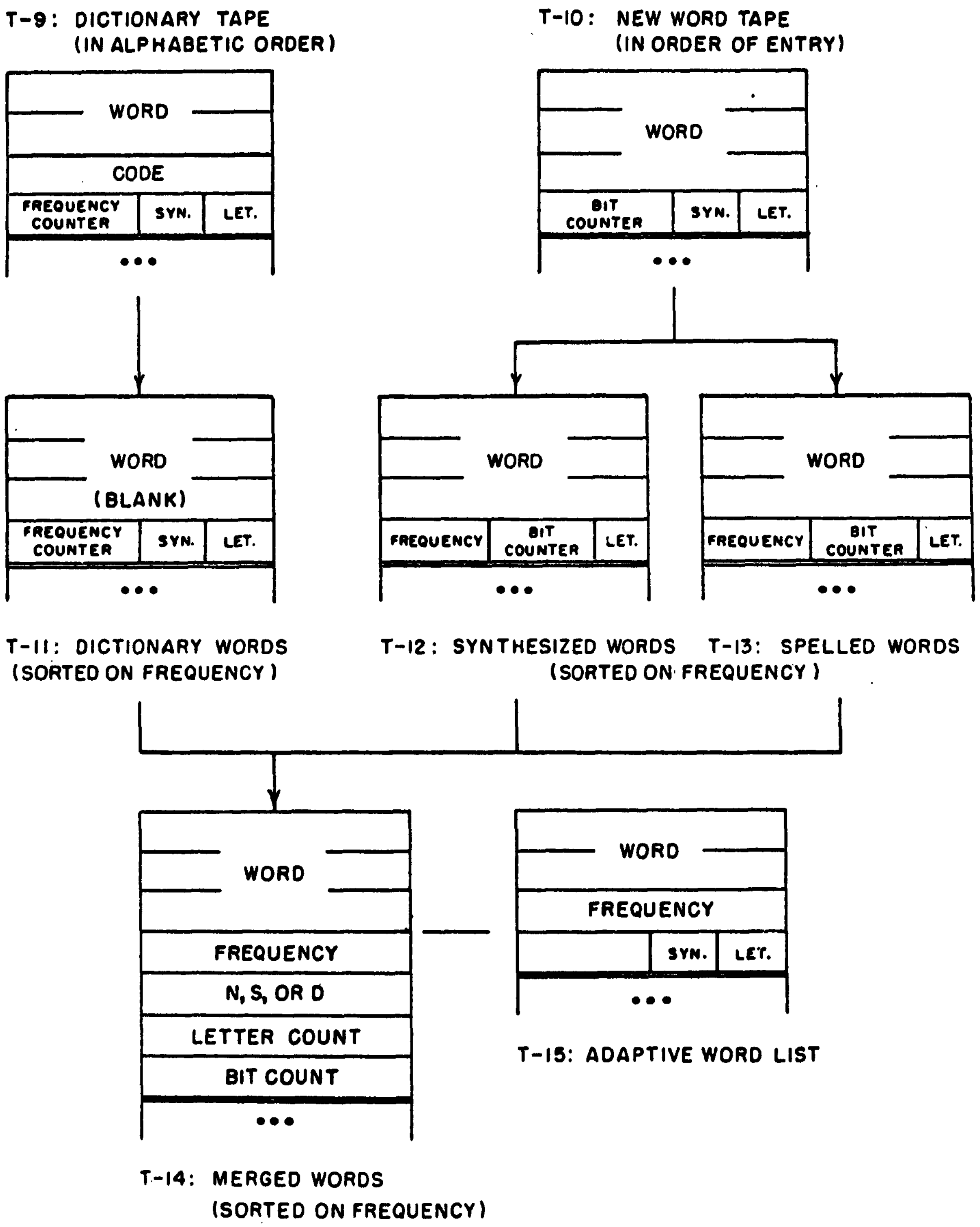


CHART 7.1

GENERAL INTERPRETATION

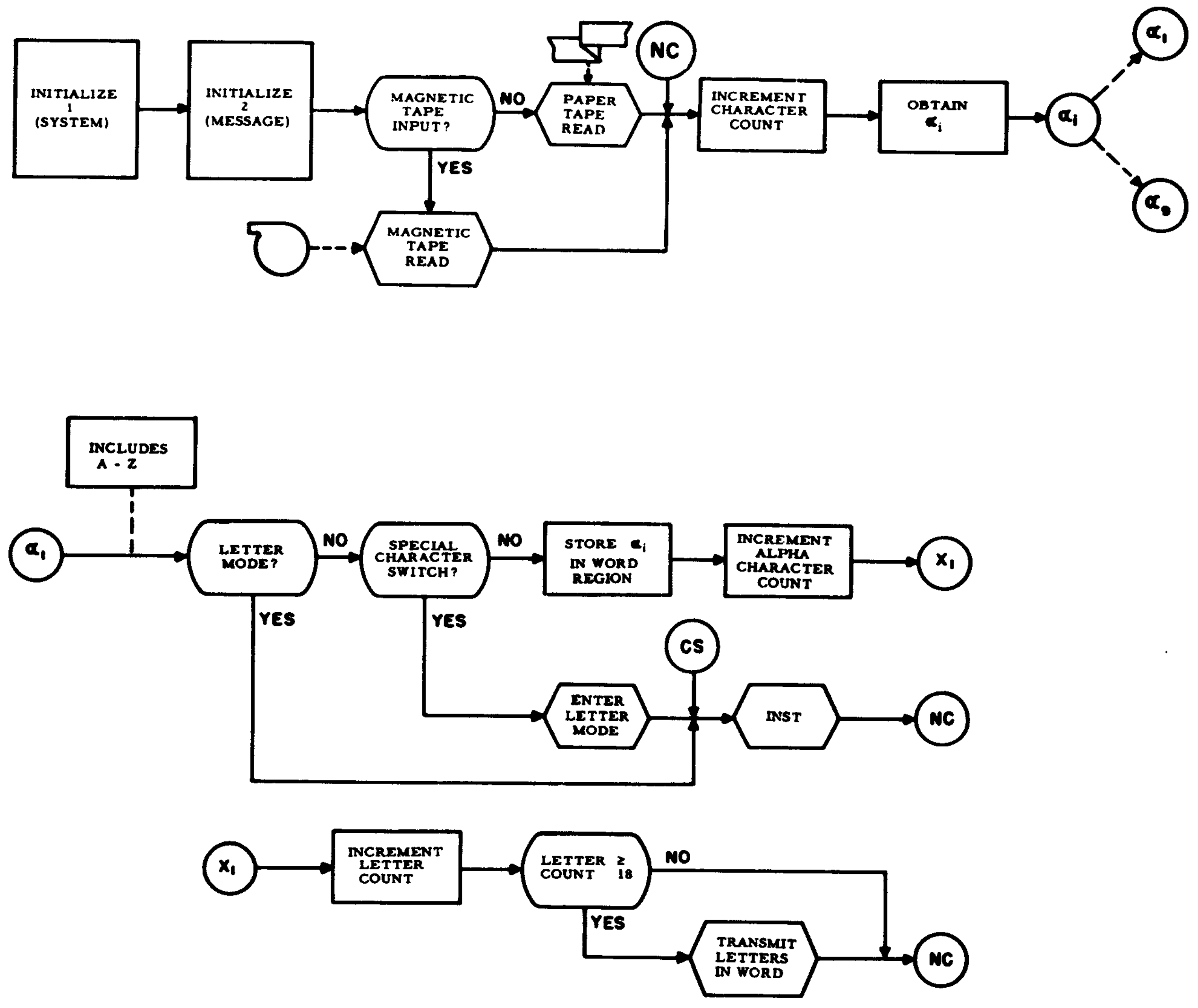


CHART 7.1 (CONTINUED)

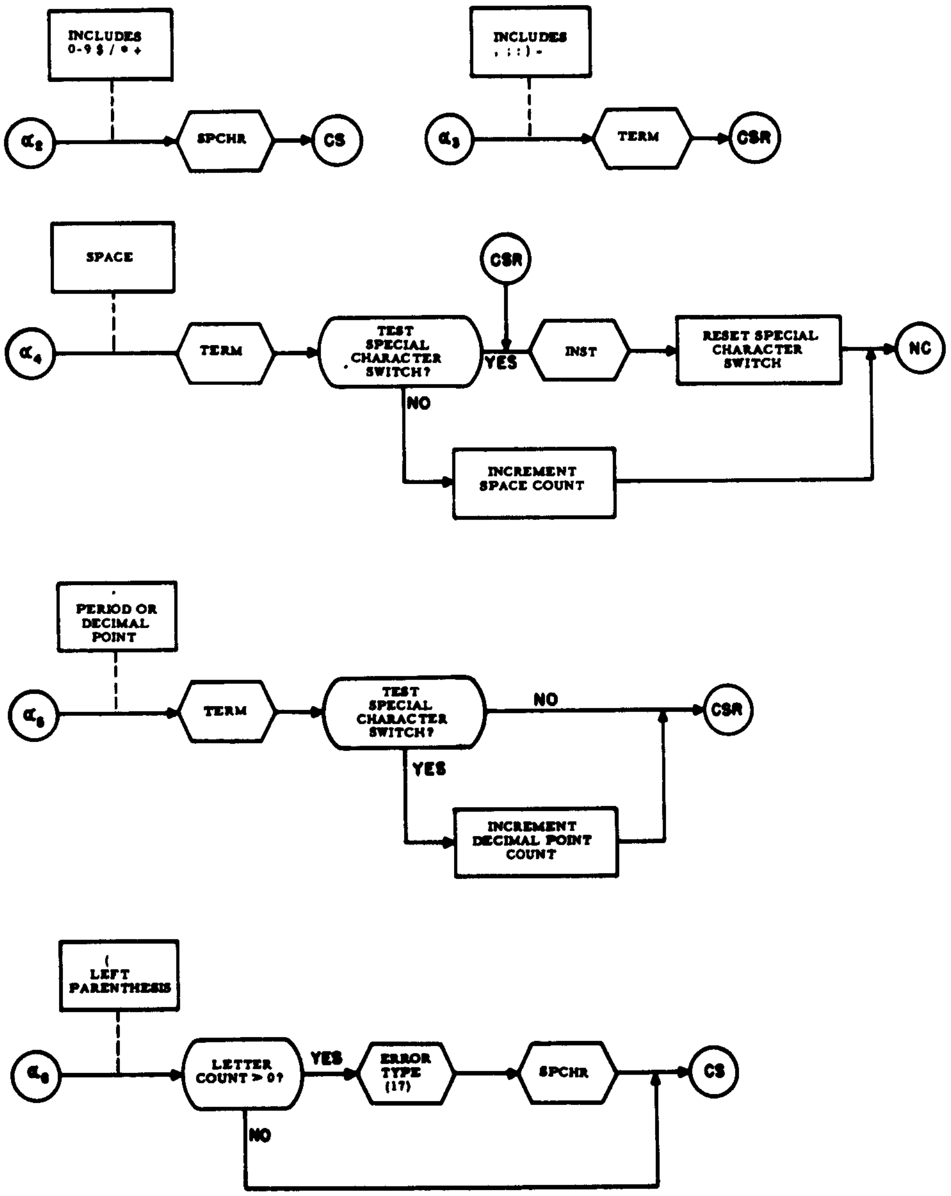


CHART 7.1 (CONTINUED)

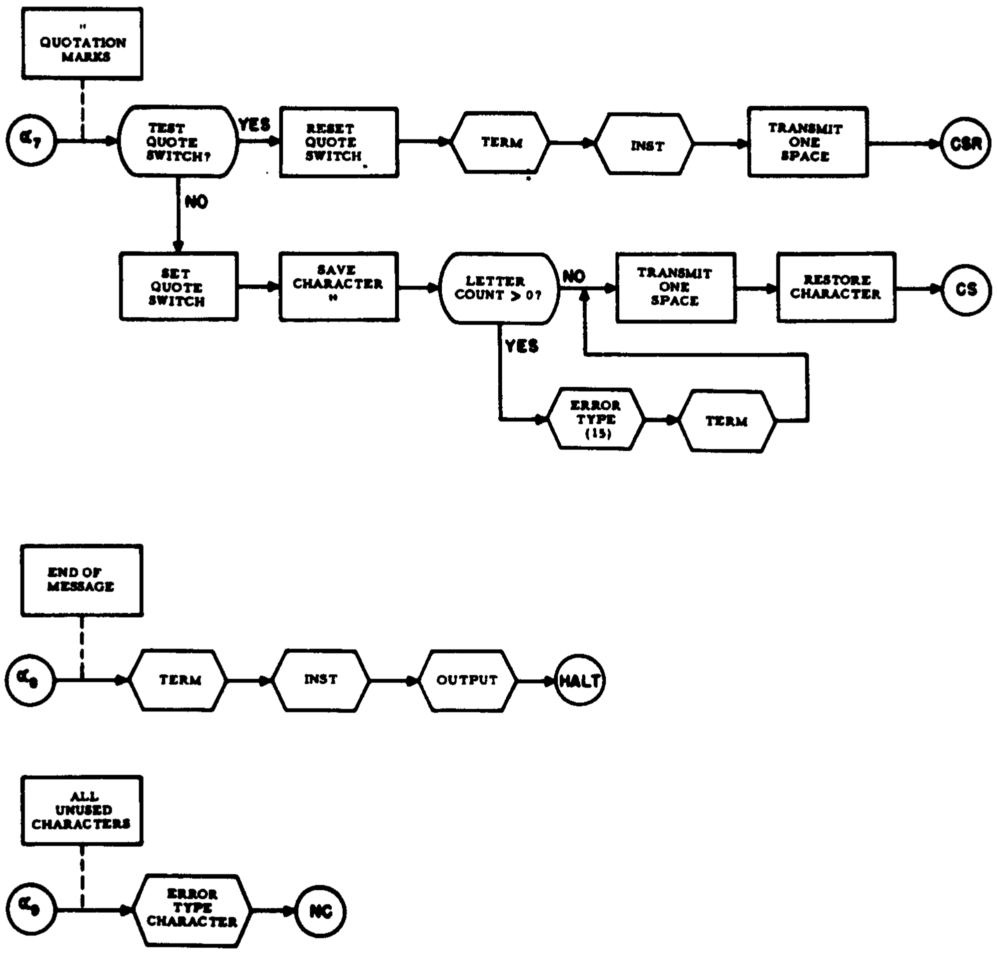


CHART 7.2

SPECIAL CHARACTER

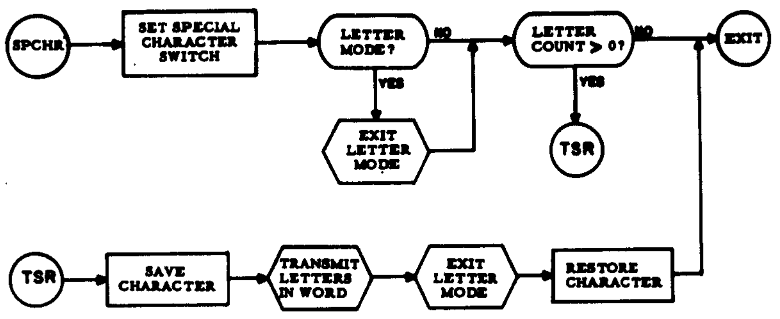


CHART 7.3

INCREMENT COUNTER

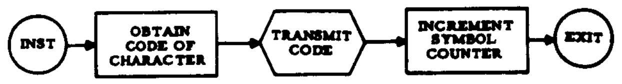


CHART 7.4

TERMINATOR

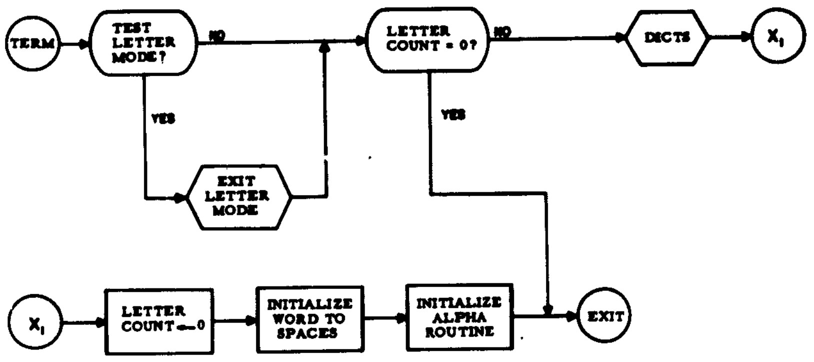


CHART 7.5
DICTIONARY SEARCH

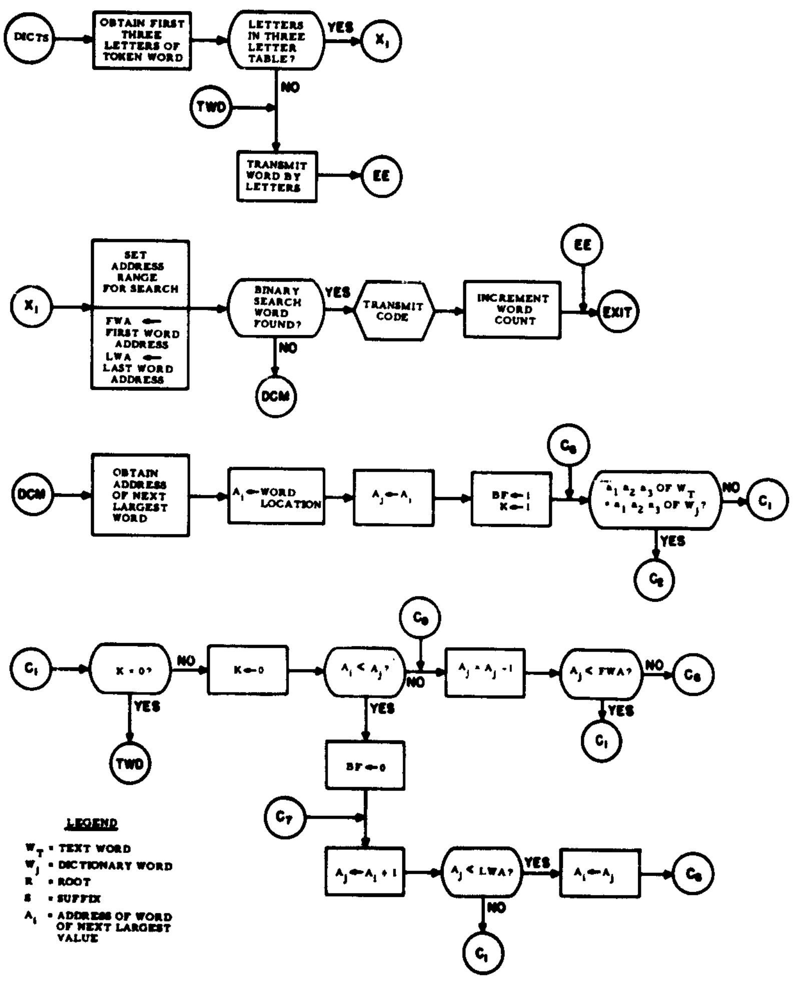


CHART 7.6

WORD DECOMPOSITION

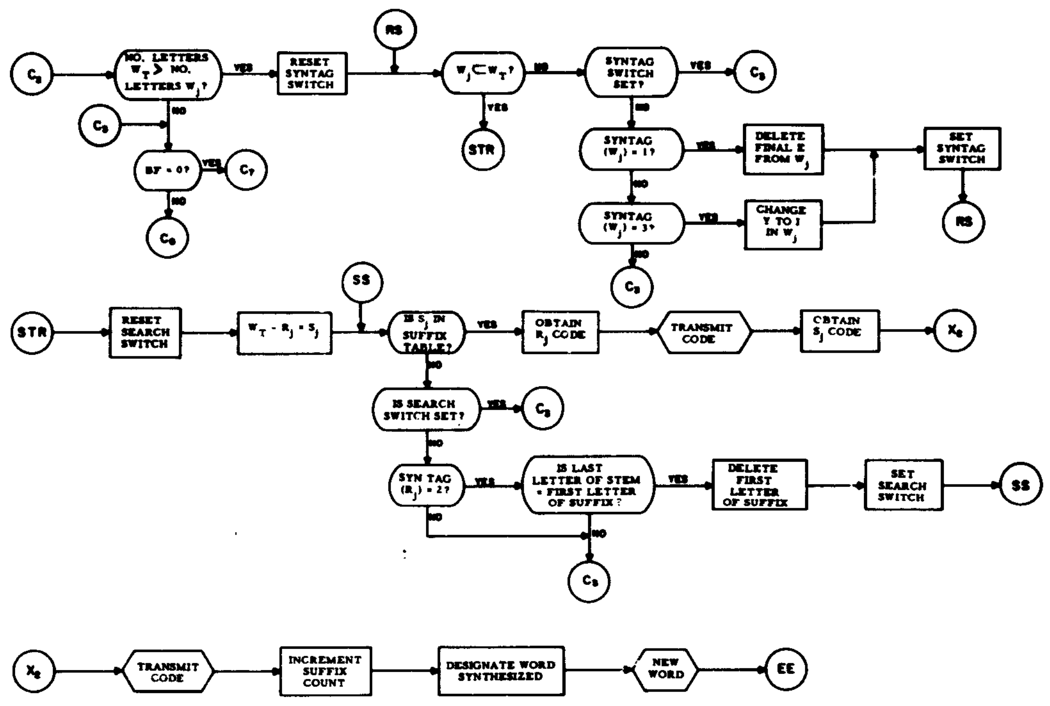


CHART 7.7
ENTER LETTER MODE

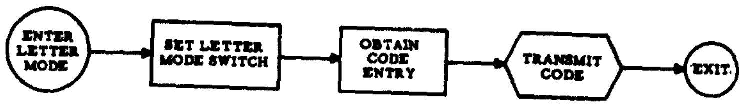


CHART 7.8
EXIT LETTER MODE

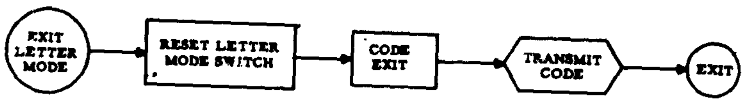


CHART 7.9
WRITE NEW WORD

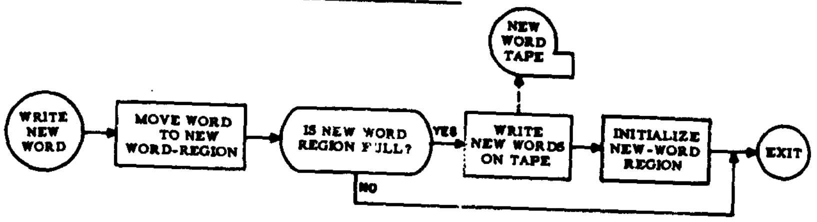


CHART 7.10
TRANSMIT LETTERS IN WORD

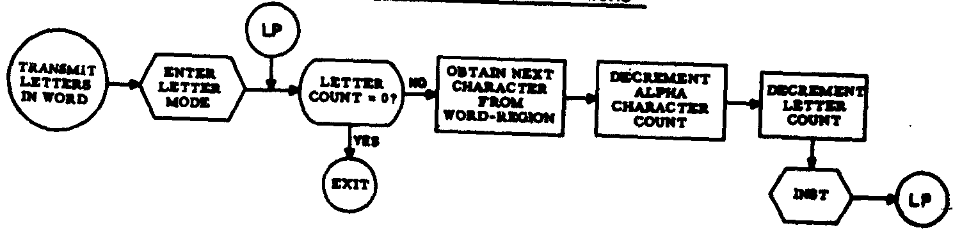


CHART 7.11
TRANSMIT WORD BY LETTERS

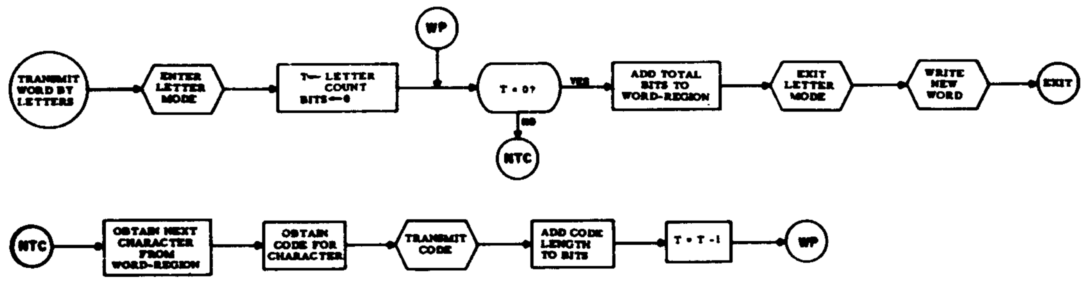


CHART 7.12
TRANSMIT CODE

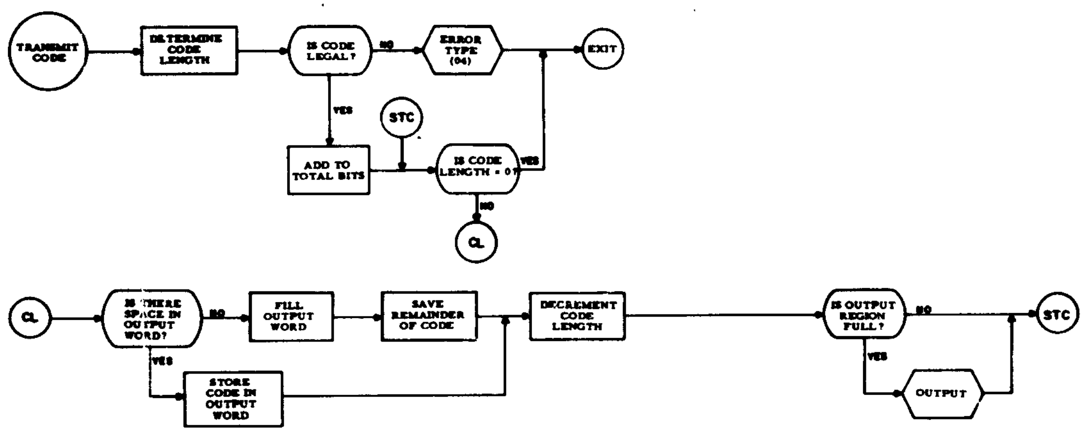


CHART 7.13

OUTPUT

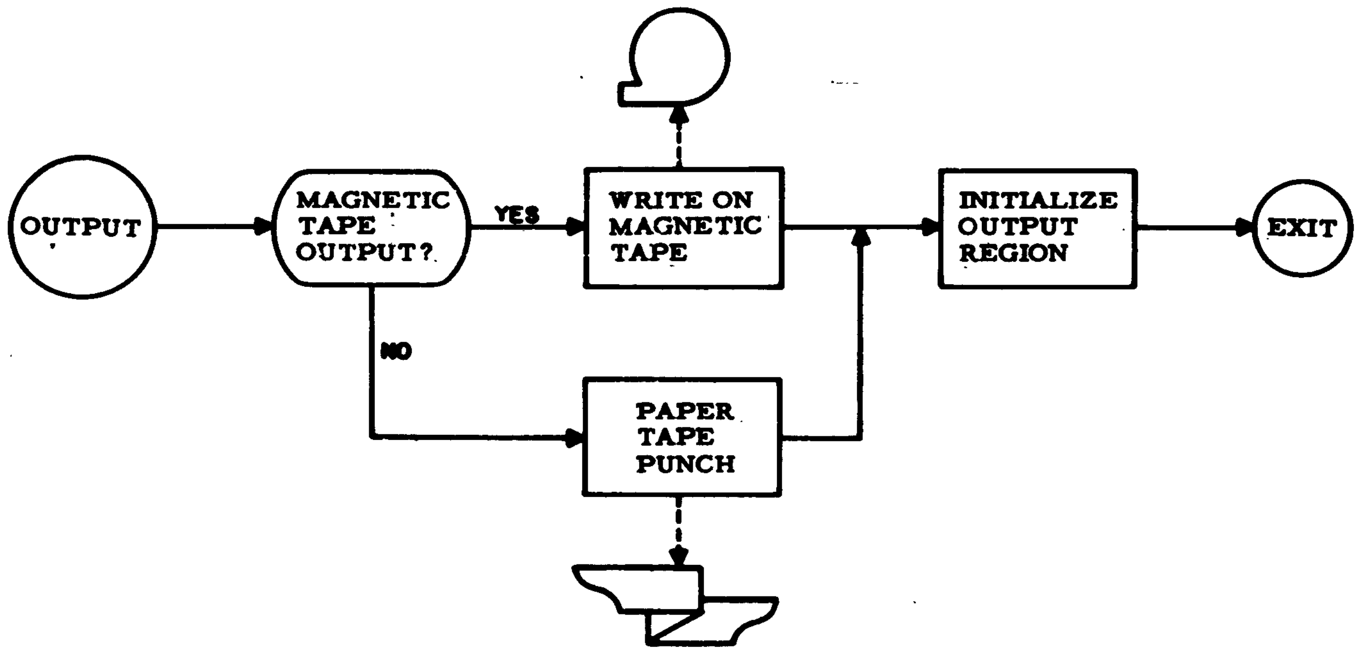


CHART 7.14

EC TRANSMIT ERROR CHECKING

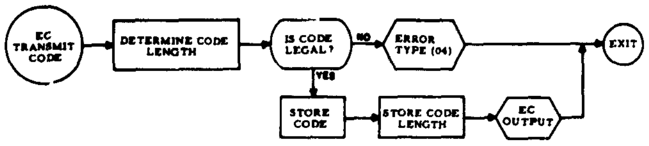


CHART 7.15

EC OUTPUT (ERROR CHECKING)

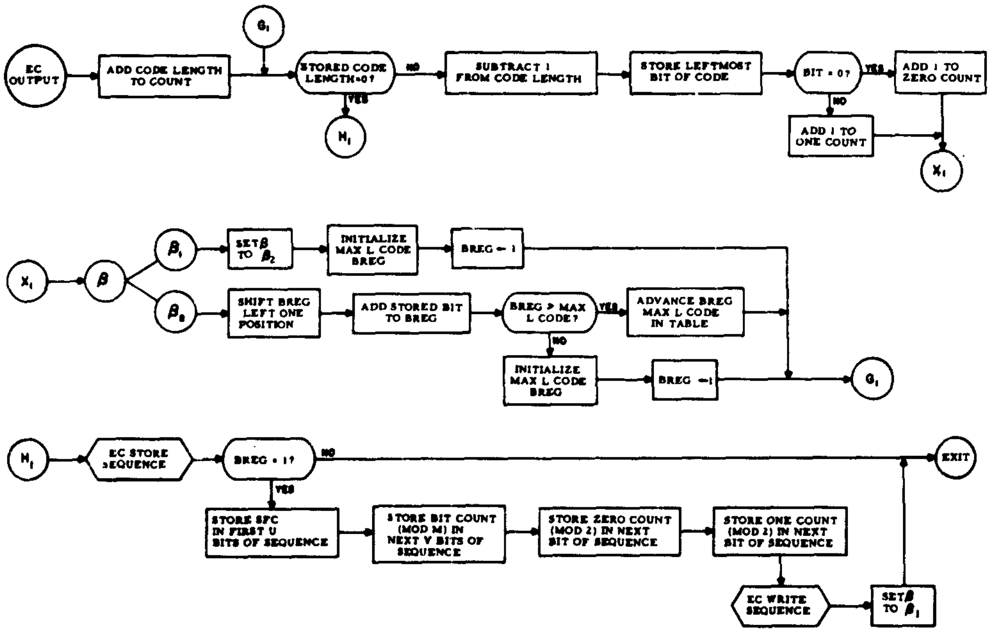

G_1
EC OUTPUT
ADD CODE LENGTH TO COUNT
STORED CODE LENGTH=0?
NO
SUBTRACT 1 FROM CODE LENGTH
STORE LEFTMOST BIT OF CODE
BIT = 0?
YES
ADD 1 TO ZERO COUNT
YES
H_1
NO
ADD 1 TO ONE COUNT
X_1
X_1
β
β_1
SET β TO β_2
INITIALIZE MAX L CODE BREG
BREG ← 1
β_2
SHIFT BREG LEFT ONE POSITION
ADD STORED BIT TO BREG
BREG > MAX L CODE?
YES
ADVANCE BREG MAX L CODE IN TABLE
NO
INITIALIZE MAX L CODE BREG
BREG ←1
G_1
H_1
EC STORE SEQUENCE
BREG = 1?
NO
EXIT
YES
STORE SFC IN FIRST U BITS OF SEQUENCE
STORE BIT COUNT (MOD M) IN NEXT V BITS OF SEQUENCE
STORE ZERO COUNT (MOD 2) IN NEXT BIT OF SEQUENCE
STORE ONE COUNT (MOD 2) IN NEXT BIT OF SEQUENCE
EC WRITE SEQUENCE
SET β TO β_1

CHART 7.16

EC STORE SEQUENCE (ERROR CHECKING)

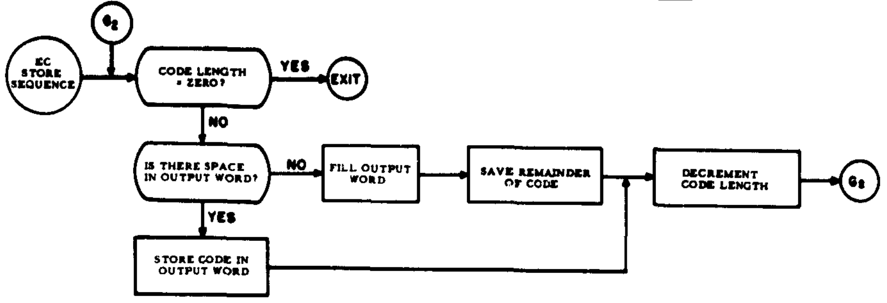


CHART 7.17

EC WRITE SEQUENCE (ERROR CHECKING)

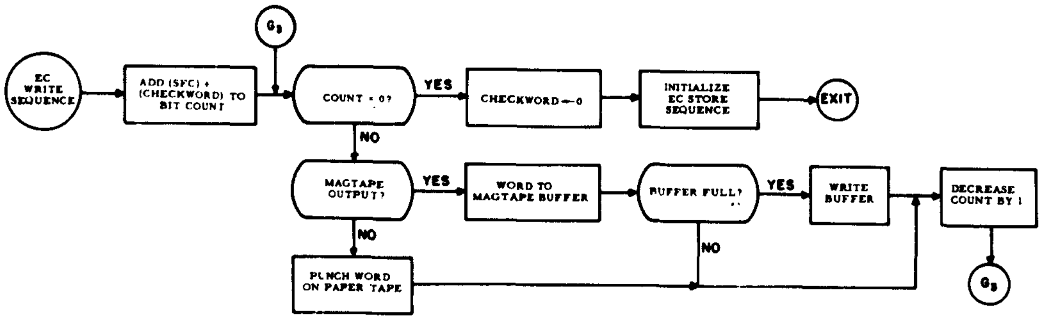


CHART 7.18

EOM FORCED SYNCHRONIZATION (ERROR CHECKING)

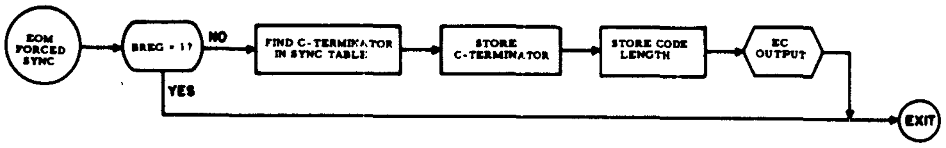
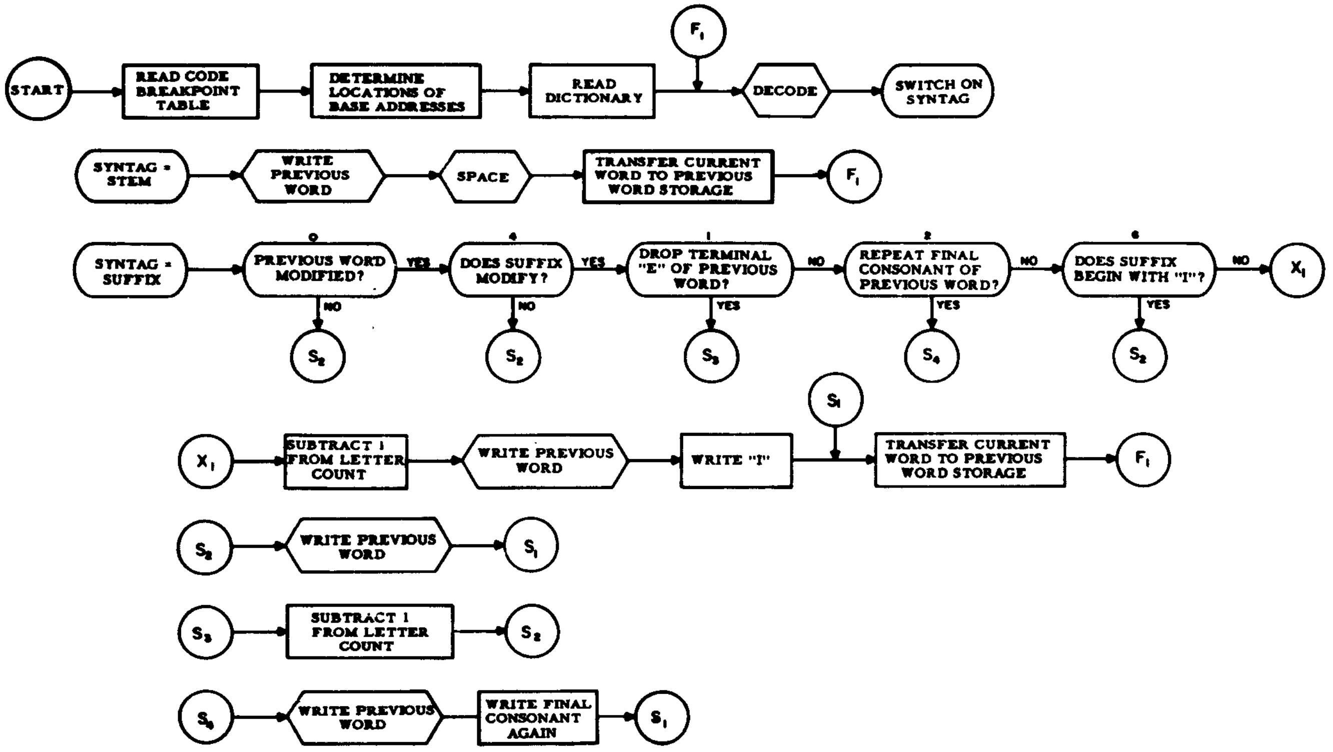


CHART 8.1

DECIPHER AND DECODE MESSAGE



START
READ CODE BREAKPOINT TABLE
DETERMINE LOCATIONS OF BASE ADDRESSES
READ DICTIONARY
F_1
DECODE
SWITCH ON SYNTAG
SYNTAG = STEM
WRITE PREVIOUS WORD
SPACE
TRANSFER CURRENT WORD TO PREVIOUS WORD STORAGE
F_1
SYNTAG = SUFFIX
0
PREVIOUS WORD MODIFIED?
YES
NO
S_2
4
DOES SUFFIX MODIFY?
YES
NO
S_2
1
DROP TERMINAL "E" OF PREVIOUS WORD?
NO
YES
S_3
2
REPEAT FINAL CONSONANT OF PREVIOUS WORD?
NO
YES
S_4
6
DOES SUFFIX BEGIN WITH "I"?
NO
YES
S_2
X_1
X_1
SUBTRACT 1 FROM LETTER COUNT
WRITE PREVIOUS WORD
WRITE "I"
S_1
TRANSFER CURRENT WORD TO PREVIOUS WORD STORAGE
F_1
S_2
WRITE PREVIOUS WORD
S_1
S_3
SUBTRACT 1 FROM LETTER COUNT
S_2
S_4
WRITE PREVIOUS WORD
WRITE FINAL CONSONANT AGAIN
S_1

CHART 8.1 (CONTINUED)

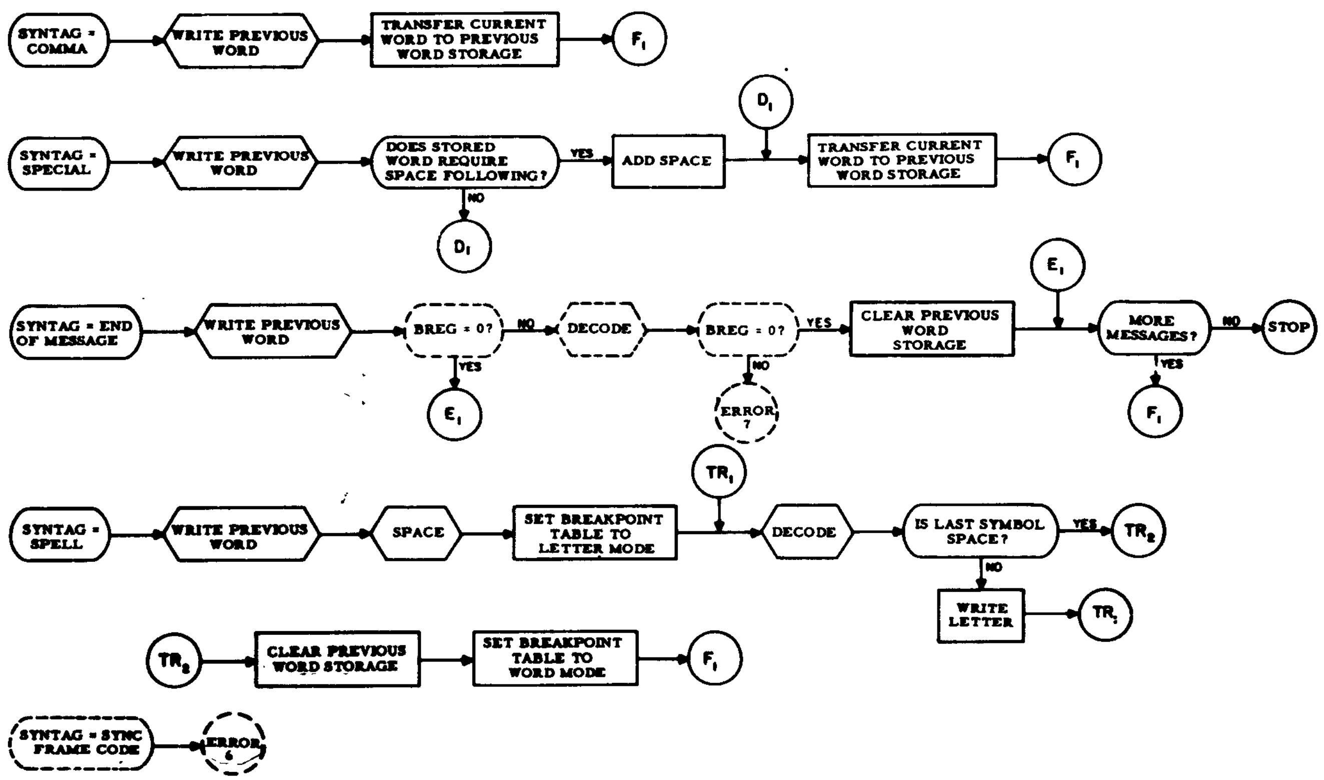


CHART 6.2

DECODE WITH ERROR DECTECTION

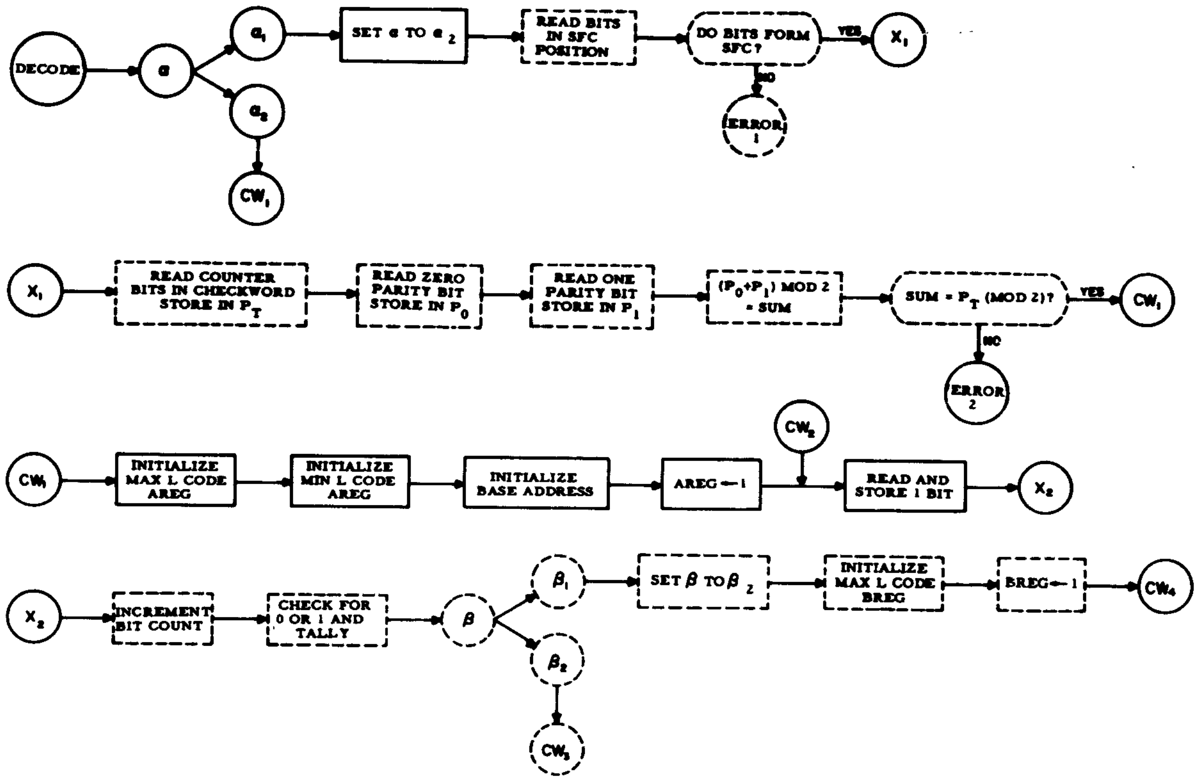
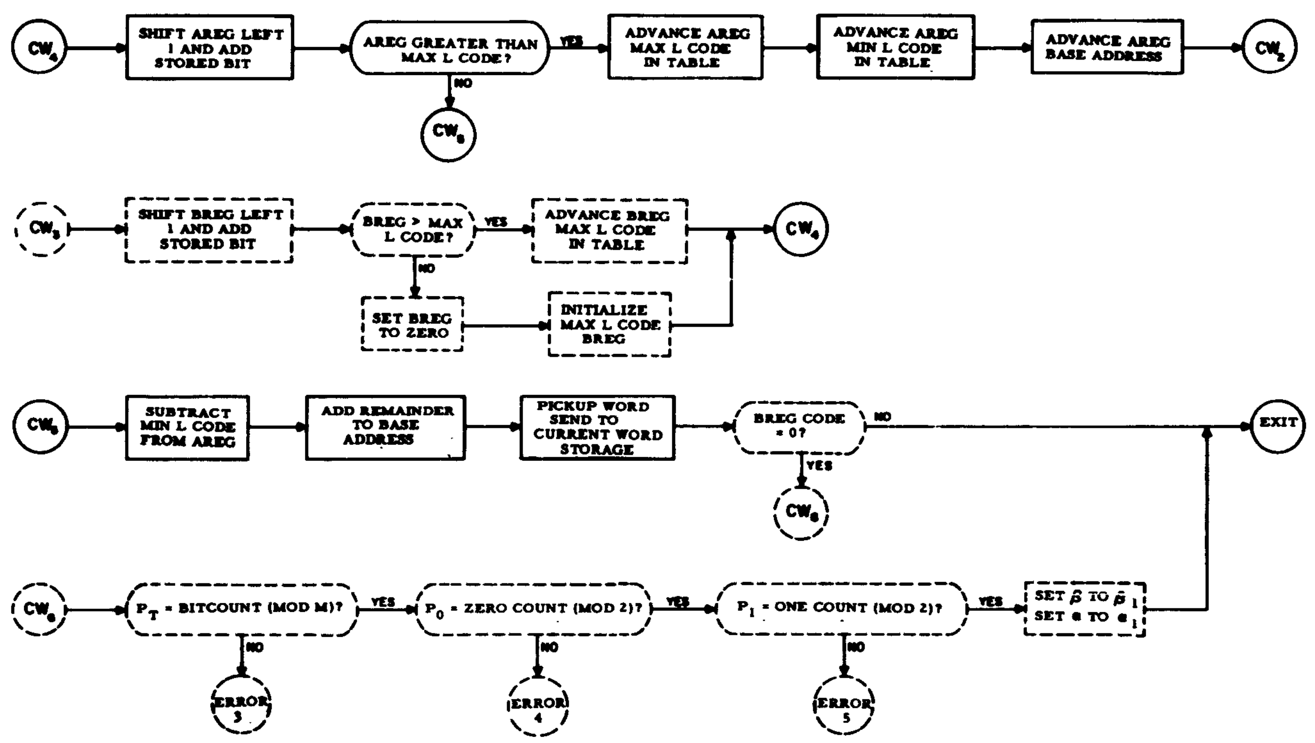

DECODE
α
α1
α2
CW1
SET α TO α 2
READ BITS IN SFC POSITION
DO BITS FORM SFC?
YES
NO
X1
ERROR 1
X1
READ COUNTER BITS IN CHECKWORD STORE IN P_T
READ ZERO PARITY BIT STORE IN P0
READ ONE PARITY BIT STORE IN P1
(P0+P1) MOD 2 = SUM
SUM = P_T (MOD 2)?
YES
NO
CW1
ERROR 2
CW1
INITIALIZE MAX L CODE AREG
INITIALIZE MIN L CODE AREG
INITIALIZE BASE ADDRESS
AREG ← 1
CW2
READ AND STORE 1 BIT
X2
X2
INCREMENT BIT COUNT
CHECK FOR 0 OR 1 AND TALLY
β
β1
β2
CW3
SET β TO β 2
INITIALIZE MAX L CODE BREG
BREG ← 1
CW4

CHART 8.2 (CONTINUED)



CW_4
SHIFT AREG LEFT 1 AND ADD STORED BIT
AREG GREATER THAN MAX L CODE?
YES
NO
CW_5
ADVANCE AREG MAX L CODE IN TABLE
ADVANCE AREG MIN L CODE IN TABLE
ADVANCE AREG BASE ADDRESS
CW_2
CW_3
SHIFT BREG LEFT 1 AND ADD STORED BIT
BREG > MAX L CODE?
YES
NO
ADVANCE BREG MAX L CODE IN TABLE
SET BREG TO ZERO
INITIALIZE MAX L CODE BREG
CW_4
CW_5
SUBTRACT MIN L CODE FROM AREG
ADD REMAINDER TO BASE ADDRESS
PICKUP WORD SEND TO CURRENT WORD STORAGE
BREG CODE = 0?
NO
YES
CW_6
EXIT
CW_6
P_T = BITCOUNT (MOD M)?
YES
NO
ERROR 3
P_0 = ZERO COUNT (MOD 2)?
YES
NO
ERROR 4
P_1 = ONE COUNT (MOD 2)?
YES
NO
ERROR 5
SET β TO β_1
SET e TO e_1

CHART 8.3
WRITE PREVIOUS WORD

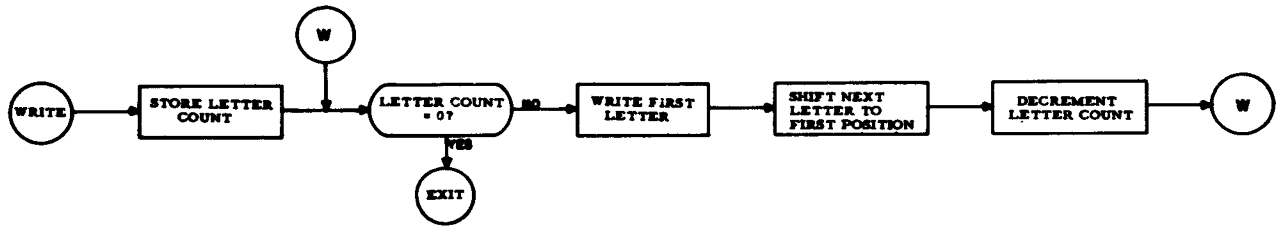


CHART 8.4
SPACE

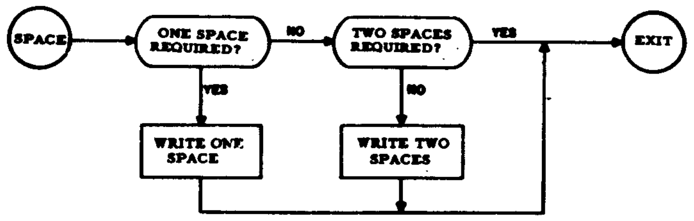
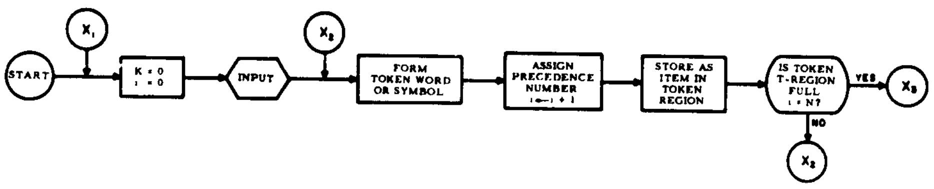


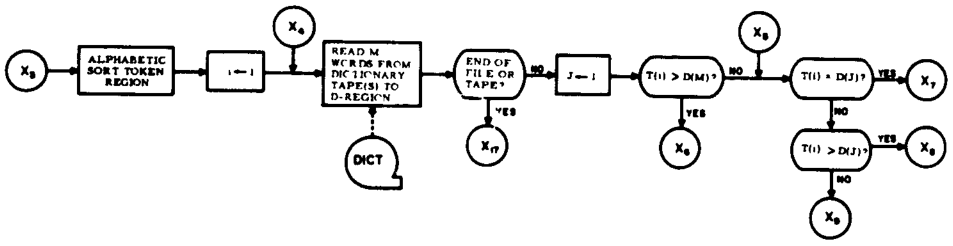
CHART 9

MAGNETIC TAPE SYSTEM WITH PRE-SORT/POST-SORT ROUTINES

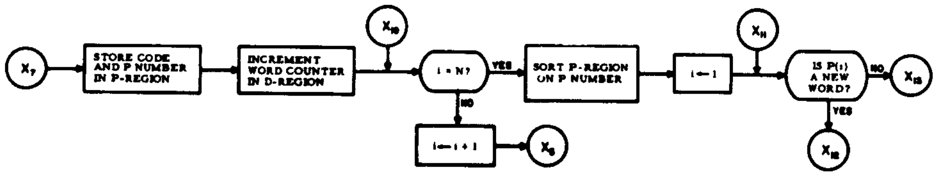
TOKEN WORD FORMATION


START
X_1
K = 0
I = 0
INPUT
X_2
FORM TOKEN WORD OR SYMBOL
ASSIGN PRECEDENCE NUMBER I←I + 1
STORE AS ITEM IN TOKEN REGION
IS TOKEN T-REGION FULL I = N?
YES
X_3
NO
X_2

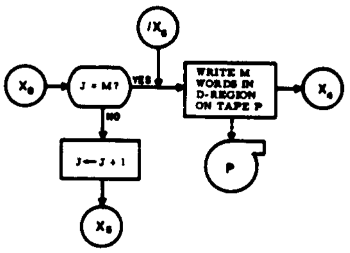
FIRST SORT, ALPHABETIC



SECOND SORT, PRECEDENCE


X_7
STORE CODE AND P NUMBER IN P-REGION
INCREMENT WORD COUNTER IN D-REGION
X_{10}
I = N?
YES
NO
I←I + 1
X_5
SORT P-REGION ON P NUMBER
I←1
X_{11}
IS P(I) A NEW WORD?
NO
X_{13}
YES
X_{12}

NEW DICTIONARY TAPE



NEW WORD

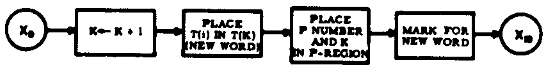
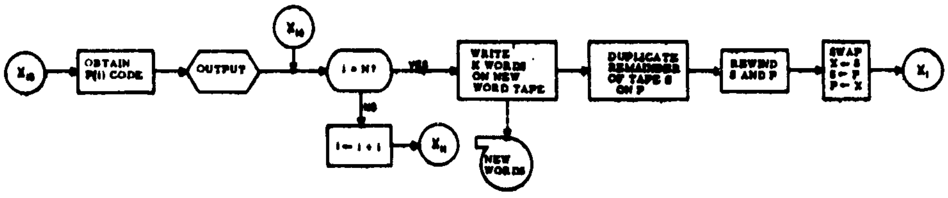
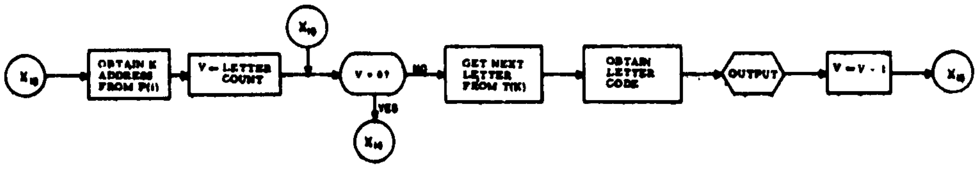

X_9
K←K + 1
PLACE T(I) IN T(K) (NEW WORD)
PLACE P NUMBER AND K IN P-REGION
MARK FOR NEW WORD
X_{10}

CHART 9 (CONTINUED)

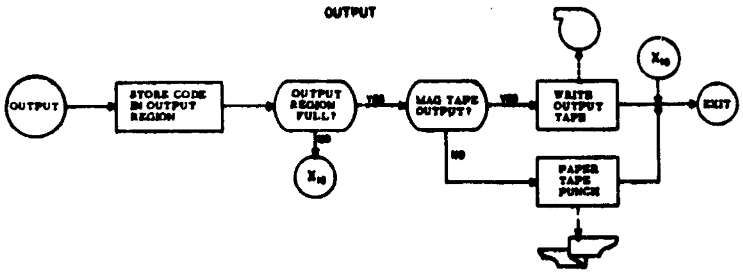
TRANSMIT CODE AND RESTART



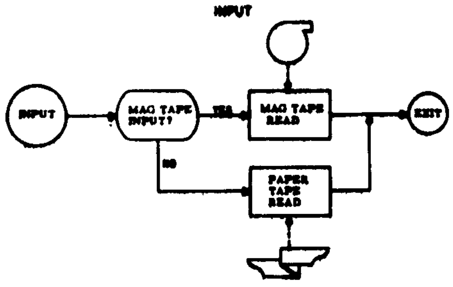
TRANSMIT LETTERS IN WORD

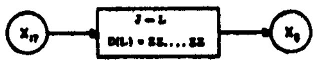


OUTPUT



INPUT





X. CONCLUSIONS

An information transmission system employing computers at both transmitter and receiver terminals has been designed and tested. An experimental dictionary consisting of 5,208 words and symbols was encoded with an efficiency of 0.996 and an information content of 1.96 binary digits per character.

A modified split dictionary format was utilized in which word types with a frequency greater than a threshold value were included in the dictionary. Thus both canonical and complex word forms were listed. In addition, 36 suffixes were listed making possible single-layered word formations. The word decomposition and synthesis programs were based upon a division of the dictionary words into four grammatical classes and the principle of the longest match was used in word searching. If an input word was not located in the experimental dictionary and could not be synthesized, it was spelled out letter-by-letter.

Compression ratios $(1 - \frac{\text{output binary digits}}{\text{input binary digits}})$ for samples of 19,710 and 50,000 word tokens were 46.84 and 53.7 percent, respectively. Information content was 3.19 and 2.78 binary digits per character.

Nearly 80 percent of the word tokens in the smaller sample were located in the dictionary as were 73.6 percent of the word tokens in the larger sample. Approximately 13 percent of the tokens were synthesized in both samples.

The effectiveness of the word synthesis routine was demonstrated by contrasting the information content of located dictionary words, synthesized, and spelled words. In the 19,710 word sample, the information content of located dictionary words was 2.76 binary digits per character compared to

3. 16 and 6. 37 binary digits per character for synthesized and spelled words. At an additional cost of only 0. 40 binary digits per character, nearly 5,000 computer words were saved in additional storage. By-passing the synthesis routine on a 1,593 word sample, dropped the compression ratio from 51. 94 to 42. 38 percent.

Eight "idiot" words were synthesized (e. g. , REST+ATE = RESTATE) without error and at a gain in transmission efficiency since it is more efficient to synthesize a word than to spell it out.

The word statistics obtained from the experimental dictionary can be used to increase the word processing rate of a computer program. The average length of 5, 153 most frequently used words was 3. 95 letters per word (exclusive of terminal space). The average length of a synthesized word was 7. 23 letters per character (exclusive of terminal space), indicating again the desirability of incorporating frequently used words in a modified split dictionary format.

Only 38 of the most frequent 5, 153 word types exceeded 12 letters in length. In a fixed word length computer, 12 letters can be established as the maximum for a stored dictionary word thus saving thousands of storage locations without an appreciable decrease in the efficiency of the system in processing words.

Using the table of ranks of both word types and word tokens according to the initial letter of a word makes possible efficient division of the dictionary between drum and core storage. The rate of 1,000 words per minute obtained at the transmitter with the UNIVAC 1105 could be increased by 30 percent by placing in available core storage those words whose first letter ranks highest in word tokens.

The word rate at the receiver where 74 percent of the most frequent word types are stored in core was approximately 1,800 words per minute.

If storage was available for the entire dictionary in the core of the UNIVAC a processing rate of 2,150 words per minute at the transmitter could be achieved. For the IBM 7090, the rate is estimated at 21,500 words per minute.

The breakdown of the dictionary into a three-letter table according to the initial three letters of the words resulted in a rapid binary search for an input word and greatly facilitated the word decomposition program.

The computation of the rank correlation between two word lists at stated values of cumulative word tokens provides a method for determining when sufficient word tokens have been processed so that an optimum encoding can be obtained.

A formula for determining the "break even" rate for cost evaluation of information transmission with a standard system as compared to one employing computers and minimum redundancy codes was developed. It has been shown that medium size computers can also be employed at a cost savings when transmission rates are high.

In addition to the information, linguistic, and programming aspects of the system, the report has included a discussion of the characteristics of exhaustive prefix encodings. It was proved that a bottom merging procedure in the Huffman encoding produces an encoding that has minimum $\sum p_i L_i$, minimum $\sum L_i$, and minimum L_{max}. Methods for determining the number of encodings for a given maximum length code and the distribution of codes in these encodings were also developed.

The error detection system based upon the property of self-synchronization of variable-length non-spaced prefix encodings shows promise of providing a very powerful technique for detecting any type of error in a message. The system would be especially effective under burst error conditions when a number of consecutive digits are deleted from a message.

The design principles set forth in this report can be translated into an effective and efficient operating system as has been demonstrated by the experimental model. Application of the design principles to a particular application would require programming for a specified computer in accordance with the flow charts that have been developed. Depending upon the storage available, decisions would be required as to the dictionary format and total number of words to be listed. Definite benefits would accrue from an enlargement of the symbol set and further sophistication of the word decomposition routine.

The interface between the computers and transmission and receiving equipment warrants further study to determine the input-output requirements and the special requirements of the error detection procedure. These studies would also be dependent upon the particular equipment selected to implement an operating system.

XI. BIBLIOGRAPHY

1. Schwartz, E. S., An Adaptive Information Transmission System Employing Minimum Redundancy Word Codes, Technical Documentary Report ASD-TDR-62-265, 1 April 1962. (AD 274 135)
2. Eldridge, R. C., Six Thousand Common English Words, Niagara Falls, 1911.
3. Dewey, G., The Relativ Frequency of English Speech Sounds, Harvard University Press, 1923.
4. Hanley, M. L., Word Index to James Joyce's 'Ulysses', Madison, Wisconsin, 1937.
5. Thorndike, E. L., and Lorge, I., The Teacher's Word Book of 30,000 Words, Columbia University, New York, 1944.
6. Miller, G. A., Newman, E. B., and Friedman, E. A., Length Frequency Statistics for Written English, Information and Control, 1: 370-389, 1958.
7. Thorndike, E. L., Teaching of English Suffixes, Columbia University, 1941.
8. Shannon, C. E., Prediction and Entropy of Printed English, Bell System Technical Journal, 30: 50-64, 1951.
9. Fano, R. M., Research Lab. for Electronics, Massachusetts Institute of Technology, Tech. Report No. 65, 1949.
10. Huffman, D. A., A Method for the Construction of Minimum Redundancy Codes, Proceedings Institute of Radio Engineers, 40: 1098-1101, 1952.
11. Gilbert, E. N. and Moore, E. F., Variable Length Binary Codings, Bell System Technical Journal, 38: 933-967, 1959.

12. Karp, R. S., Minimum-Redundancy Coding for the Discrete Noiseless Channel, Trans. I. R. E., IT-7: 27-38, 1961.

13. Gorn, S., Common Programming Language Task, Final Report No. AD60URI, Part I, Moore School of Cooperative Research, University of Pennsylvania, June, 1960.

14. Mandelbrot, B., On Recurrent Noise Limiting Coding, Symposium on Information Networks, pp. 205-221, New York, April, 1954.

15. Hibbard, T. N., Some Combinatorial Properties of Certain Trees with Applications to Searching and Sorting, Journal ACM, 9: 13-28, 1962.

16. Brillouin, L., Science and Information Theory, Academic Press, New York, 1956.

17. Neumann, P. G., Efficient Error-Limiting Variable-Length Codes, Trans. I. R. E., IT-8: 292-304, July, 1962.

18. Bemer, R. W., Do It by the Numbers, Communications Association for Computing Machinery, 3: 530, 1960.

19. Golomb, S. W., Gordon, B., and Welch, L. R., Comma-free Codes, Canadian Journal Math., 10: 202-209, 1958.

20. Gilbert, E. N., Synchronization of Binary Messages, Trans. I. R. E., IT-6: 470-477, 1960.

21. Moore, E. F., Gedanken-experiments on Sequential Machines, in Automata Studies, ed. Shannon, C. E. and McCarthy, J., Princeton University Press, 1956, pp. 129-153.

22. Flores, I., Analysis of Internal Computer Sorting, Journal ACM, 8: 41-80, 1961.

UNCLASSIFIED

Aeronautical Systems Division, Dir/ Avionics, Electromagnetic Warfare and Communications Lab., Wright-Patterson Air Force Base, Ohio.
Rpt. Nr. ASD-TDR-62-265, PART II.
AN ADAPTIVE INFORMATION TRANSMISSION SYSTEM EMPLOYING MINIMUM REDUNDANCY WORD CODES, by E. S. Schwartz. Final Report, Phase 2, 1 June 1963. 241 p. incl. illus. tables, charts, 22 refs. (Contract AF 33(616)-7882). Unclassified Report.
An experimental model of an adaptive information transmission system employing minimum redundancy word codes is

1. Adaptive Systems
2. Information System
3. Communications
4. Language
5. Coding

UNCLASSIFIED

UNCLASSIFIED

Aeronautical Systems Division, Dir/ Avionics, Electromagnetic Warfare and Communications Lab., Wright-Patterson Air Force Base, Ohio.
Rpt. Nr. ASD-TDR-62-265, PART II.
AN ADAPTIVE INFORMATION TRANSMISSION SYSTEM EMPLOYING MINIMUM REDUNDANCY WORD CODES, by E. S. Schwartz. Final Report, Phase 2, 1 June 1963. 241 p. incl. illus. tables, charts, 22 refs. (Contract AF 33(616)-7882). Unclassified Report.
An experimental model of an adaptive information transmission system employing minimum redundancy word codes is

1. Adaptive Systems
2. Information System
3. Communications
4. Language
5. Coding

UNCLASSIFIED

UNCLASSIFIED

described. Flow charts of computer programs for generating a canonical encoding, preparing transmitter and receiver dictionary tapes, adapting the dictionary to usage, and for enciphering, deciphering, and decoding messages are presented. Compression ratios for processed messages were determined. A procedure for determining when dictionary changes are required is outlined. Characteristics of exhaustive prefix encodings are investigated. An error-detection system based upon the property of self-synchronization of variable length non-spaced prefix encodings is presented and experimental results are given.

I. AFSC Project 4335; Task 433519
II. Contract AF 33(616)-7882
III. Armour Research Foundation, Chicago, Illinois
IV. E. S. Schwartz
V. ARF Project E153
VI. not aval. for OTS
VII. In ASTIA collection

UNCLASSIFIED

UNCLASSIFIED

described. Flow charts of computer programs for generating a canonical encoding, preparing transmitter and receiver dictionary tapes, adapting the dictionary to usage, and for enciphering, deciphering, and decoding messages are presented. Compression ratios for processed messages were determined. A procedure for determining when dictionary changes are required is outlined. Characteristics of exhaustive prefix encodings are investigated. An error-detection system based upon the property of self-synchronization of variable length non-spaced prefix encodings is presented and experimental results are given.

I. AFSC Project 4335; Task 433519
II. Contract AF 33(616)-7882
III. Armour Research Foundation, Chicago, Illinois
IV E. S. Schwartz
V. ARF Project E153
VI. not aval. for OTS
VII. In ASTIA collection

UNCLASSIFIED

UNCLASSIFIED

Aeronautical Systems Division, Dir/
Avionics, Electromagnetic Warfare and
Communications Lab., Wright-Patterson
Air Force Base, Ohio.
Rpt. Nr. ASD-TDR-62-265, PART II.
AN ADAPTIVE INFORMATION
TRANSMISSION SYSTEM EMPLOYING
MINIMUM REDUNDANCY WORD CODES,
by E. S. Schwartz. Final Report, Phase
2, 1 June 1963. 241 p. incl. illus. tables,
charts, 22 refs. (Contract AF 33(616)-
7882). Unclassified Report.
An experimental model of an adaptive
information transmission system employing
minimum redundancy word codes is

1. Adaptive Systems
2. Information System
3. Communications
4. Language
5. Coding

UNCLASSIFIED

UNCLASSIFIED

described. Flow charts of computer programs for generating a canonical encoding, preparing transmitter and receiver dictionary tapes, adapting the dictionary to usage, and for enciphering, deciphering, and decoding messages are presented. Compression ratios for processed messages were determined. A procedure for determining when dictionary changes are required is outlined. Characteristics of exhaustive prefix encodings are investigated. An error-detection system based upon the property of self-synchronisation of variable length non-spaced prefix encodings is presented and experimental results are given.

I. AFSC Project 4335; Task 433519
II. Contract AF 33(616)-7882
III. Armour Research Foundation, Chicago, Illinois
IV. E. S. Schwartz
V. ARF Project E153
VI. not aval. for OTS
VII. In ASTIA collection

UNCLASSIFIED

UNCLASSIFIED

Aeronautical Systems Division, Dir/
Avionics, Electromagnetic Warfare and
Communications Lab., Wright-Patterson
Air Force Base, Ohio.
Rpt. Nr. ASD-TDR-62-265, PART II.
AN ADAPTIVE INFORMATION
TRANSMISSION SYSTEM EMPLOYING
MINIMUM REDUNDANCY WORD CODES,
by E. S. Schwartz. Final Report, Phase
2, 1 June 1963. 241 p. incl. illus. tables,
charts, 22 refs. (Contract AF 33(616)-
7882). Unclassified Report.
An experimental model of an adaptive
information transmission system employing
minimum redundancy word codes is

1. Adaptive Systems
2. Information System
3. Communications
4. Language
5. Coding

UNCLASSIFIED

UNCLASSIFIED

described. Flow charts of computer programs for generating a canonical encoding, preparing transmitter and receiver dictionary tapes, adapting the dictionary to usage, and for enciphering, deciphering, and decoding messages are presented. Compression ratios for processed messages were determined. A procedure for determining when dictionary changes are required is outlined. Characteristics of exhaustive prefix encodings are investigated. An error-detection system based upon the property of self-synchronisation of variable length non-spaced prefix encodings is presented and experimental results are given.

I. AFSC Project 4335; Task 433519
II. Contract AF 33(616)-7882
III. Armour Research Foundation, Chicago, Illinois
IV. E. S. Schwartz
V. ARF Project E153
VI. not aval. for OTS
VII. In ASTIA collection

UNCLASSIFIED